# Predicting molten steel endpoint temperature using a feature-weighted model optimized by mutual learning cuckoo search

Qiangda Yang [1*], Jie Zhang [2], Zhi Yi [1]

[1] *School of Metallurgy, Northeastern University, Shenyang 110819, China*

[2] *School of Engineering, Newcastle University, Newcastle upon Tyne NE1 7RU, UK*

** Corresponding author e-mail: yangqd@smm.neu.edu.cn*

**Abstract:** A feature-weighted neural network model for the prediction of the endpoint temperature of molten steel (MSET) in a ladle furnace (LF) is proposed in this paper. Accurate prediction of MSET is essential for promoting product quality, reducing production costs and enhancing productivity. Considering that different features have different impacts on the MSET during the process of LF refining, a weight is applied to each feature before feeding the feature to neural networks. A mutual learning cuckoo search (MLCS) algorithm is proposed to simultaneously determine the feature weights and network parameters of the proposed prediction model. The search of each cuckoo in the basic cuckoo search algorithm and many of its variants is performed independently, which may decrease the algorithms' performance. The proposed MLCS algorithm introduces two new search strategies, the mutual learning-based search strategy and the bottom reinforcement learning-based search strategy. The superior performance of MLCS is first confirmed with 20 benchmark optimization problems. Then, MLCS is applied to optimize the feature weights and network parameters in the feature-weighted MSET prediction model. Application to modeling the production data from a 300 t LF in an iron & steel plant in China demonstrates the effectiveness of the proposed feature-weighted neural network model.

**Keywords:** modeling; cuckoo search; feature weighed; ladle furnace; molten steel temperature.

## 1. Introduction

LF is an important equipment for secondary refining in iron and steel industries [1]. Tight control of the MSET in LF is essential to improve product quality, reduce production costs and enhance productivity [2, 3]. However, in the actual production process, there are at present no proper hardware sensors to continuously measure the molten steel temperature, which brings difficulty to the precise control of MSET [4]. Thus, more and more research is trying to resolve this problem by developing MSET prediction models in LF.

Traditionally, molten steel temperature prediction models are developed on the basis of the energy conservation law and thermodynamics [5, 6]. However, since there are over-idealized assumptions and hard-to-obtain parameters, mechanistic models are unable to be utilized efficiently for accurate prediction [3, 7]. Whereas in data-driven modeling, the model is developed exclusively from the historical process input–output data [8], generally not involving with over-idealized assumptions and hard-to-obtain parameters. As a result, many researchers have attempted to predict the temperature of molten steel in LF by various data-driven modeling methods. For instance, Sun *et al.* [9] use a neural network to predict the temperature. Tian *et al.* [10, 11] propose to predict the temperature by a new incremental learning modeling method and a modified AdaBoost.RT algorithm-based ensemble ELM modeling method. Wang *et al.* [12] present a temperature prediction model combining the regression tree algorithm and general regression neural networks.

Most of the aforementioned data-driven methods treat all features (i.e. inputs) equally in the process of model development. However, for actual LF refining processes, each feature has different influence on the MSET [13]. Accordingly, we draw on the feature-weighted concept in the field of categorization [14] and propose a feature-weighted data-driven modeling method for the development of the MSET prediction model in this paper. Single-hidden layer feed-forward

neural networks (SLFNs) are capable of approximating any nonlinear function [15, 16] and have been successfully applied in modeling many LF refining processes to predict the molten steel temperature [7, 9-11, 17]. Therefore, in this study, we utilize this type of networks for the development of the MSET prediction model.

Global optimization techniques need to be used in the determination of feature weights and network parameters (i.e. the connection weights and thresholds of the corresponding SLFN) of the proposed feature-weighted MSET prediction model (referred to as FW-SLFN model). Cuckoo search (CS) is a nature-inspired optimization algorithm proposed by Yang and Deb [18] through emulating the brood parasitism behavior of some cuckoo species. As a global optimization algorithm with some advantages like relatively good balance between the local search (exploitation) and global search (exploration), simplicity, and efficiency [19, 20], CS has been successfully applied to numerous optimization problems in various fields with remarkable performance, such as model parameter optimization [21-24], controller design [25, 26], optimal placement design [27, 28], benchmark function optimization [29], robotic assembly sequence planning [30], online route planning [31], hydrothermal scheduling problems [32], and system optimization [33, 34]. Besides, some comparative analyses of CS with several widely-used nature-inspired optimization algorithms have proved that CS is overall more successful than these algorithms [35, 36]. Motivated by these encouraging results, in this study we employ CS for the joint optimization of the weight of each feature and the network parameters. However, CS is also vulnerable to slow and premature convergence and its search capability should be further enhanced [20, 37].

Researchers have developed a series of CS variants. Huang et al. propose five chaos-enhanced cuckoo search (CCS) algorithms, in which chaotic sequences are utilized to enhance the initialized nest location, change the step size of Lévy flight and reset the location of nest beyond

the boundary, and numerical results show that among the five CCSs the one with Gauss map for the generation of chaotic sequences achieves the best performance [37]. Naik and Panda present an adaptive cuckoo search (ACS) algorithm in which the step size is made adaptive from the knowledge of the fitness value and current location [38]. Valian et al. come up with an improved cuckoo search (ICS) algorithm that dynamically changes the values of the step length scaling factor and alien egg discover probability [39]. Wang et al. introduce a nearest neighbour cuckoo search (NNCS) algorithm with a nearest neighbour strategy to choose guides to search for new solutions and a probabilistic mutation strategy to control the solutions learn from the nearest neighbour solutions in partial dimensions instead of all of them, and the experimental results indicate that the fitness-based nearest neighbour strategy is better than the solution-based strategy [40]. Cheung et al. present a new CS variant with quantum mechanism based nonhomogeneous search strategies, termed nonhomogeneous cuckoo search (NoCuSa) [41]. The above-mentioned CS variants have enhanced the search accuracy or convergence speed from different aspects. Nevertheless, there still is a drawback in the basic CS and these variants: the information (search experience) exchange among cuckoos is lacking that would considerably impact the search performance. In order to address this issue, we present a mutual learning CS (MLCS) in this paper for the purpose of enhancing the exchange of search experience among cuckoos, and thereby, the search capability of the algorithm. In MLCS, each cuckoo is afforded an opportunity to acquire the search experience of an exemplar chosen from the population, which is enlightened by a consensus that it would be of great help for a group to efficiently fulfill a task if the members could learn from each other. In addition, a bottom reinforcement learning-based search strategy is also introduced in the presented MLCS algorithm to further increase its performance. Finally, MLCS is applied to optimize the weight of each feature together with the network parameters so as to accomplish the development of the presented FW-SLFN model.

The main contribution of this paper includes the following three aspects:

(1) Considering that each feature has different influence on the MSET in actual LF refining processes, a feature-weighted SLFN-based modeling method for the development of the MSET prediction model (i.e. FW-SLFN model) is proposed.

(2) A new CS variant, MLCS, is presented to enhance the exchange of search experience among cuckoos, and thereby, the search performance. In MLCS, a novel mutual learning-based search strategy is proposed to give each cuckoo a chance to acquire the search experience of a selected exemplar. Besides, a bottom reinforcement learning-based search strategy is also proposed to further increase the search capability of the algorithm. The experimental results on some benchmark functions are promising and confirm the excellent search performance of the proposed MLCS algorithm.

(3) Based on actual production data from a 300 t LF in a Chinese iron & steel plant, MLCS is applied to optimize the feature weights and network parameters in the proposed FW-SLFN model, and the results indicate the effectiveness of this prediction model and the necessity of the feature-weighted strategy.

The remaining parts of the paper are organized as follows. Section 2 gives a brief introduction of the LF refining process, presents the main factors influencing the MSET, and describes the development framework of the proposed FW-SLFN model. Section 3 briefly discusses the basic CS, details the new MLCS algorithm, and validates its performance by applying to 20 well-known benchmark functions. In Section 4, the implementation of MLCS for solving the joint optimization problem of the feature weights and network parameters of the FW-SLFN model is described. In Section 5, based on actual production data from a 300 t LF in an iron & steel plant in China, test results of the proposed prediction model are provided and again the performance of MLCS is demonstrated. Finally, conclusions are drawn and considerations for

future works are pointed out in Section 6.

## 2. FW-SLFN model

### *2.1. Process flow of LF refining*

LF refining technology has an essential role to play in the secondary metallurgic process. It can adjust the temperature and component of molten steel and improve the purity of molten steel by operations like arc heating, slagging, alloying, argon blowing, etc. Additionally, it can greatly improve the buffering capacity between steelmaking and continuous casting. The whole process flow of LF refining, as schematically shown in Fig.1, starts from the entry of the ladle and ends at the exit of the ladle. To be specific, when the ladle containing molten steel enters the heating station, the operations of sampling and temperature measuring, argon blowing, furnace cover lowering, slag adding, power on for arc heating and power off are carried through in turn. After that multiple sampling and temperature measuring are often required to obtain the satisfied endpoint temperature and component of molten steel. In any adjacent sampling period, similar operations are executed in the sequence of alloy adding and slag adding, power supply, and power off. In the end, after the final sampling and temperature measuring, the furnace cover is lifted, the argon blowing cube is removed, and then the ladle is carted out from the heating station to accomplish this refining process.

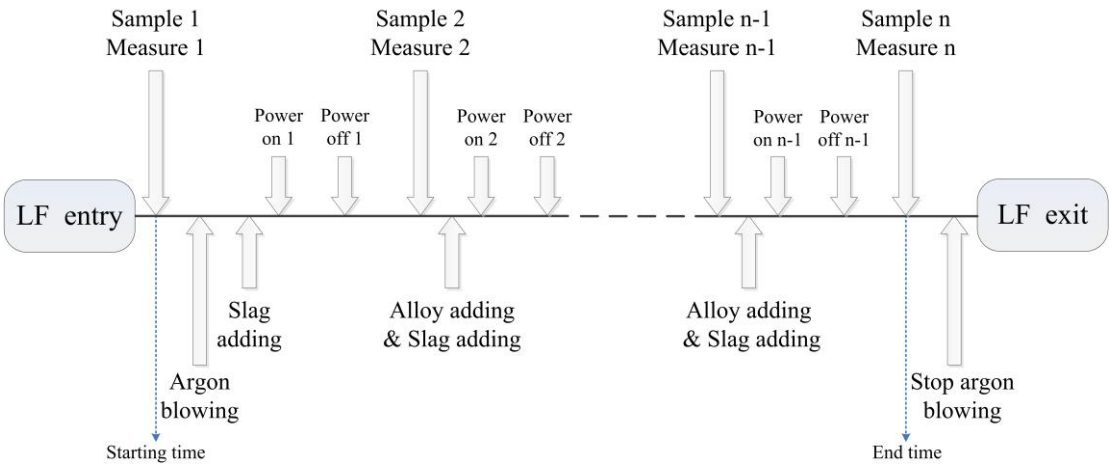


**Fig. 1.** Process flow of LF refining.

### *2.2. Influence factors of MSET*

The influence factors of MSET are selected on the basis of energy equilibrium during the LF refining process, as shown in Fig. 2. To be specific, if a LF is taken as a system, the input energy ($Q_{in}$), the output energy ($Q_{out}$), and the energy absorbed by the system ($Q_{abs}$) should balance, as formulated below:

$$Q_{in} = Q_{out} + Q_{abs} \tag{1}$$

where $Q_{in}$ is the heat gain from arc heating ($Q_{arc}$); $Q_{out}$ is composed of four parts, including the heat taken away by argon ($Q_{arg}$), the heat loss from the top surface ($Q_{srf}$) that can be reflected by the energy change of cooling water in the water-cooled cover, the heat loss from the ladle lining ($Q_{lnn}$) that can be reflected by the ladle heat status and refining time, and the heat effect of additions ($Q_{add}$); and $Q_{abs}$ is the heat absorbed by molten steel ($Q_{stl}$) used to make its temperature rise.

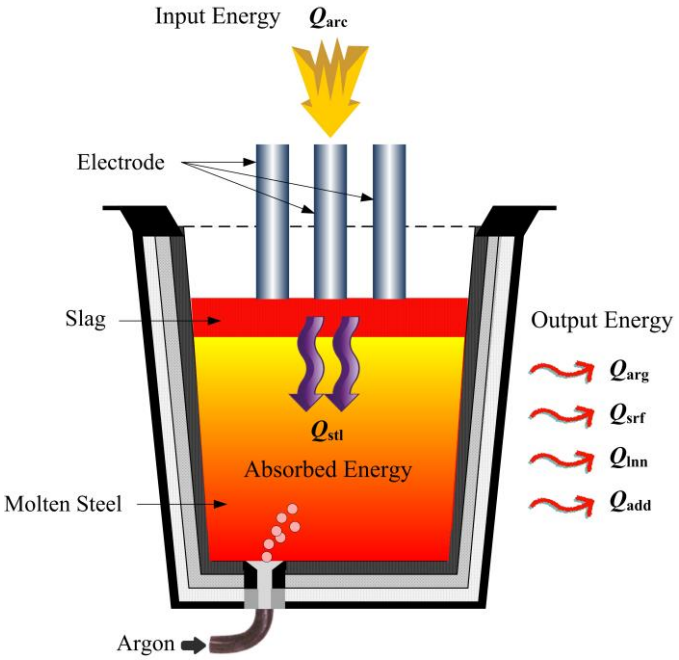


**Fig. 2.** Energy equilibrium diagram of LF.

According to the energy equilibrium analysis above, it can be seen that there are mainly eight factors determining the MSET in a LF. They are the refining power consumption, the amount of argon blowing, the energy change of cooling water, the ladle heat status, the refining time, the heat effect of additions, the initial molten steel temperature, and the molten steel weight. For the sake of brevity, these eight factors are denoted as $ftr_1$, $ftr_2$, $ftr_3$, $ftr_4$, $ftr_5$, $ftr_6$, $ftr_7$, and $ftr_8$ in this paper, respectively.

### *2.3. Basic framework of the FW-SLFN model*

As mentioned in the introduction, different factors have different influences on MSET in actual LF refining processes [13]. By utilizing this characteristic of LF refining processes, it may be possible to enhance the performance of our MSET prediction model. To this end, we present the feature-weighted data-driven modeling method, which assigns each feature a different weight to reflect its different influence on the MSET. For the presented method, one of the keys is the determination of feature weights. In this study, we determine them together with the network

parameters simultaneously, as described below.

Mathematically, the proposed feature-weighted data-driven MSET prediction model (i.e. the FW-SLFN model) can be formulated as

$$MSET=\varphi(w_1 ftr_1, w_2 ftr_2, \cdots, w_8 ftr_8, \boldsymbol{P}_N) \tag{2}$$

where $\varphi$ denotes the mapping of the inputs to the output of the FW-SLFN model ; $w_1$, $w_2$, ⋯, $w_8$ are the weights of $ftr_1$, $ftr_2$, ⋯, $ftr_8$, respectively; and $\boldsymbol{P}_N$ is the vector of connection weights and thresholds of the SLFN. Fig. 3 shows the structure of this prediction model schematically, where *bias* = –1 is designed for the introduction of thresholds.

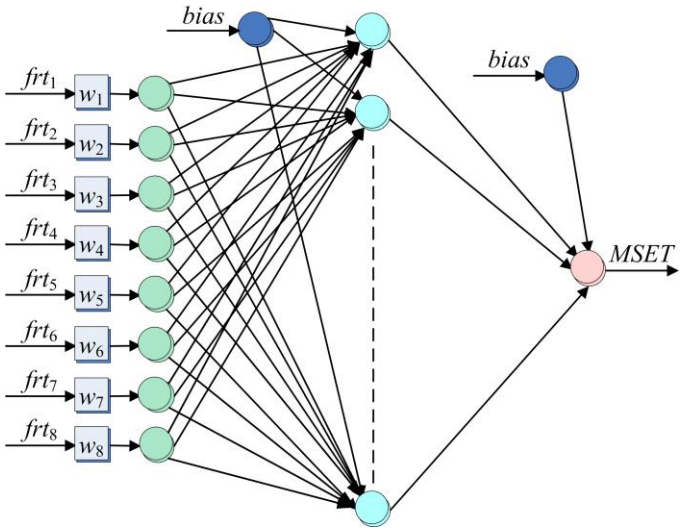


**Fig. 3.** Schematic representation of the FW-SLFN model.

To obtain the presented FW-SLFN model, two parameter vectors, namely the feature weight vector $\boldsymbol{w}$ ($\boldsymbol{w}$ = [$w_1$, $w_2$, ⋯, $w_8$]) and the network parameter vector $\boldsymbol{P}_N$, need to be determined. In this study, the optimal values of $\boldsymbol{w}$ and $\boldsymbol{P}_N$ are obtained by using the new MLCS algorithm. To be specific, $\boldsymbol{w}$ and $\boldsymbol{P}_N$ are regarded as the parameter vectors to be identified in the model described by Eq. (2); then their values are simultaneously determined by using MLCS to minimize the following objective function

$$J(\boldsymbol{w}, \boldsymbol{P}_{\mathrm{N}}) = \frac{\sum_{m=1}^{M} \left| x_m - \hat{x}_m(\boldsymbol{w}, \boldsymbol{P}_{\mathrm{N}}) \right|}{M} \tag{3}$$

where $M$ is the number of MSET samples utilized for model development; and $x$ and $\hat{x}$ denote the measured and predicted MSET values, respectively. More details about how to determine $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ will be given in Section 4.

## 3. Optimization algorithm

### *3.1. Basic CS*

Cuckoo search is a nature-inspired optimization algorithm designed on the basis of the phenomenon that certain species of cuckoos lay their eggs in other birds' nests together with the Lévy flights behavior of some birds and fruit flies. For simplicity, Yang and Deb [18] utilize the following three idealized rules to describe the cuckoo search process: the first is that every time each cuckoo lays one egg and puts the egg into a randomly selected nest; the second is that the nests with high quality of eggs are carried over to the next generation; and the last is that the number of available host nests is changeless and the host bird can discover an alien egg laid by a cuckoo with a probability $p_a \in [0, 1]$. In such a situation, the host bird can either throw the egg away or just give up the nest, and then construct a new nest. According to the above rules, the basic CS has been implemented with two main methods to update the solutions. In addition, it should be pointed out that a cuckoo, an egg and a nest are all equivalent to a solution, and only minimization problems are considered in the remainder of this paper without loss of generality.

In the first component of the basic CS, Lévy flights are used to generate the new solution $\boldsymbol{z}_i^{\mathrm{new}}$ for each cuckoo $i$ according to the following equation:

$$\boldsymbol{z}_i^{\mathrm{new}} = \boldsymbol{z}_i + \alpha \cdot \boldsymbol{s}_{\mathrm{L}} \oplus (\boldsymbol{z}_i - \boldsymbol{z}_{\mathrm{FT}}) \oplus \boldsymbol{r} \tag{4}$$

where $\alpha$ is a step length scaling factor, and it can be set to 0.01 for most cases [42, 43]; the symbol $\oplus$ means entry-wise multiplications; $\boldsymbol{r}$ is a vector with elements obeying the standard normal distribution $N$ (0, 1); $\boldsymbol{z}_{\mathrm{FT}}$ represents the fitness-based top cuckoo, namely the cuckoo with the top (i.e. lowest) fitness value in the current population; and $\boldsymbol{s}_{\mathrm{L}}$ is a step length vector that obeys the Lévy distribution. By the Mantegna algorithm [44], $\boldsymbol{s}_{\mathrm{L}}$ can be generated as given below:

$$s_{\mathrm{L}} = \frac{u}{|v|^{1/\beta}} \tag{5}$$

where $u$ and $v$ are random variables with the following normal distributions

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \tag{6}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\cdot \sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\cdot \beta \cdot 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1 \tag{7}$$

where $0 < \beta \le 2$ and $\beta = 1.5$ in the basic CS [42]; and $\Gamma$ denotes the Gamma function.

In the second component of the basic CS, new solutions are generated through imitating the discovery of alien eggs, and mathematically it can be formulated as:

$$\boldsymbol{z}_i^{\mathrm{new}} = \boldsymbol{z}_i + rnd_1 \cdot \boldsymbol{H}(\boldsymbol{p_a} - \boldsymbol{rnd}_2) \oplus (\boldsymbol{z}_j - \boldsymbol{z}_k) \tag{8}$$

where $\boldsymbol{z}_j$ and $\boldsymbol{z}_k$ are two randomly selected solutions; $rnd_1$ is a random number generated from a uniform distribution between 0 and 1, hereafter referred to as $U$ (0, 1); $\boldsymbol{p_a}$ is a vector with all elements being $p_a$; $\boldsymbol{rnd}_2$ is a random vector with all elements generated from $U$ (0, 1); and $\boldsymbol{H}(\bullet)$ is a step function, defined as

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{H}(x_1, x_2, \cdots, x_d, \cdots, x_D) = [H_1, H_2, \cdots, H_d, \cdots, H_D] \tag{9}$$

where $H_d = 1$ if $x_d > 0$, otherwise $H_d = 0$; $D$ is the search space dimension.

From Eqs. (4) and (8) and the description above, the search process of the basic CS can be summarize as follows: firstly, within lower and upper limits (if any), $S$ number of cuckoos (eggs, nests or solutions) are randomly initialized, where $S$ is the population size. Next, each cuckoo $\boldsymbol{z}_i$ is assessed by its fitness value and then the basic CS enters the iterative process. The evolutionary purpose of each iteration is the same, namely to search for new solutions by applying the above-mentioned two components of the basic CS in sequence. After each component, the comparison of fitness values between each old solution and each new solution at the same nest will be performed and the solutions with lower fitness values are retained. The best solution (i.e. $\boldsymbol{z}_{\mathrm{FT}}$) is updated after each iteration and it is selected as the optimal solution of the search process. More details on the basic CS are given in [42].

### *3.2. Proposed MLCS*

As stated in the introduction, the basic CS and many of its variants lack of experience exchange among cuckoos, which could lead to impaired performance. In order to overcome this problem, this paper proposes a new CS variant called MLCS, which is mainly designed on the basis of a consensus that it would be greatly helpful for a group to efficiently fulfill a task if the members could learn from each other. Compared with the basic CS, MLCS also includes two iterative components, but it introduces two new search strategies. The first is the mutual learning-based (ML-based) search strategy which allows each cuckoo to have an opportunity to learn from an exemplar selected from its population. The second is the bottom reinforcement learning-based (BRL-based) search strategy which provides an extra opportunity for the cuckoo in the bottom of the population (referred to as $\boldsymbol{z}_{\mathrm{B}}$) to learn from two top cuckoos, that is, (1) the fitness-based top cuckoo (i.e. $\boldsymbol{z}_{\mathrm{FT}}$) and (2) the fitness and spatial location-based top cuckoo (referred to as $\boldsymbol{z}_{\mathrm{FST}}$). In

the remaining parts of this section, the two new search strategies are explained first and then the overall framework of MLCS is presented.

*3.2.1. ML-based search strategy*

This strategy aims to enhance the experience interaction among cuckoos, and thereby, the algorithm's performance. To this end, it allows each cuckoo *i* to have an opportunity to learn from an exemplar, and accordingly modifies Eqs. (4) and (8) into Eqs. (10) and (11), respectively. And, being similar to the basic CS, Eqs. (10) and (11) are respectively used as the main evolutionary equations of the two components of MLCS.

$$\boldsymbol{z}_i^{\text{new}} = \boldsymbol{z}_i + \Delta\boldsymbol{z}_i^{\text{L}} \oplus (\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{c}_1 - \boldsymbol{rnd}_3)) + \Delta\boldsymbol{z}_i^{\text{M}_1} \oplus \boldsymbol{H}(\boldsymbol{c}_1 - \boldsymbol{rnd}_3) \tag{10}$$

$$\boldsymbol{z}_i^{\text{new}} = \boldsymbol{z}_i + \boldsymbol{H}(\boldsymbol{p}_a - \boldsymbol{rnd}_2) \oplus (\Delta\boldsymbol{z}_i^{\text{A}} \oplus (\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{c}_1 - \boldsymbol{rnd}_4)) + \Delta\boldsymbol{z}_i^{\text{M}_2} \oplus \boldsymbol{H}(\boldsymbol{c}_1 - \boldsymbol{rnd}_4)) \tag{11}$$

where

$$\Delta\boldsymbol{z}_i^{\text{L}} = 0.01 \cdot \boldsymbol{s}_{\text{L}} \oplus (\boldsymbol{z}_i - \boldsymbol{z}_{\text{FT}}) \oplus \boldsymbol{r} \tag{12}$$

$$\Delta\boldsymbol{z}_i^{\text{M}_1} = c_2 \cdot \boldsymbol{rnd}_5 \oplus (\boldsymbol{z}_{e_i} - \boldsymbol{z}_i) \tag{13}$$

$$\Delta\boldsymbol{z}_i^{\text{A}} = rnd_1 \cdot (\boldsymbol{z}_j - \boldsymbol{z}_k) \tag{14}$$

$$\Delta\boldsymbol{z}_i^{\text{M}_2} = c_2 \cdot \boldsymbol{rnd}_6 \oplus (\boldsymbol{z}_{e_i} - \boldsymbol{z}_i) \tag{15}$$

In the above equations, $\boldsymbol{I}$ is a vector with all elements being one; $\boldsymbol{c}_1$ is a vector with all elements being $c_1$, which is a coefficient used to control the new solution percentages produced by the Lévy flight (or the alien egg discovery) and by the mutual learning; $\boldsymbol{rnd}_3$, $\boldsymbol{rnd}_4$, $\boldsymbol{rnd}_5$, and $\boldsymbol{rnd}_6$ are four random vectors with all their elements generated from $U$ (0, 1); $c_2$ is the learning coefficient; $\boldsymbol{H}(\bullet)$ is the same step function as defined in Eq. (9); and $\boldsymbol{z}_{ei}$ is cuckoo *i*'s exemplar. In this paper,

it is identified as follows: firstly three candidates are selected randomly from the population $\boldsymbol{z}_1,\cdots,\boldsymbol{z}_{i-1},\boldsymbol{z}_{i+1},\cdots,\boldsymbol{z}_S$ (note that $\boldsymbol{z}_i$ itself is not included); then the one with the lowest fitness value that exists in the selected three candidates is identified as $\boldsymbol{z}_{ei}$, and meanwhile its index is assigned to $e_i$.

*3.2.2. BRL-based search strategy*

This strategy is designed to further increase the search performance of MLCS. To achieve this purpose, the two top cuckoos, namely $\boldsymbol{z}_{\mathrm{FT}}$ and $\boldsymbol{z}_{\mathrm{FST}}$ are used to reinforce the learning of $\boldsymbol{z}_{\mathrm{B}}$, which is formulated as

$$\boldsymbol{z}_{\mathrm{B}}^{\mathrm{new}} = \boldsymbol{z}_{\mathrm{B}} + c_2 \cdot \boldsymbol{rnd}_7 \oplus (\boldsymbol{z}_{\mathrm{FT}} - \boldsymbol{z}_{\mathrm{B}}) + c_2 \cdot \boldsymbol{rnd}_8 \oplus (\boldsymbol{z}_{\mathrm{FST}} - \boldsymbol{z}_{\mathrm{B}}) \tag{16}$$

where $\boldsymbol{rnd}_7$ and $\boldsymbol{rnd}_8$ are two random vectors with all their elements generated from $U(0, 1)$. $\boldsymbol{z}_{\mathrm{B}}$ is identified on the basis of its fitness value, and the cuckoo with the bottom (i.e. highest) fitness value that exists in the population is assigned to $\boldsymbol{z}_{\mathrm{B}}$. As mentioned, $\boldsymbol{z}_{\mathrm{FT}}$ is also identified on the basis of its fitness value and the cuckoo with the top (i.e. lowest) fitness value that exists in the population is assigned to $\boldsymbol{z}_{\mathrm{FT}}$. As for $\boldsymbol{z}_{\mathrm{FST}}$, the fitness value and the spatial location are both considered to identify it. More specially, a metric related to the fitness value and spatial location of each cuckoo (referred as to FS) is calculated first and then $\boldsymbol{z}_{\mathrm{FST}}$ are identified. To calculate the FS metric for each cuckoo $i$, the weight $W(\boldsymbol{z}_i)$ which depends on the fitness value, is calculated first by the following equations:

$$w(\boldsymbol{z}_i) = \frac{f_{\max} - f(\boldsymbol{z}_i)}{f_{\max} - f_{\min}} \tag{17}$$

$$W(\boldsymbol{z}_i) = \frac{w(\boldsymbol{z}_i)}{\sum_i^S w(\boldsymbol{z}_i)} \tag{18}$$

where $w(\boldsymbol{z}_i)$ is cuckoo $i$'s weight value before normalization; $f(\boldsymbol{z}_i)$ is cuckoo $i$'s fitness value; and $f_{\max}$ and $f_{\min}$ are respectively the maximum and minimum fitness values that exist in the current population. Next, the FS metric for each cuckoo $i$ (referred as to $FS(\boldsymbol{z}_i)$) is calculated by weighting its Euclidean distance to the cuckoo with the lowest fitness value in the population (i.e. $\boldsymbol{z}_{\mathrm{FT}}$) according to its $W(\boldsymbol{z}_i)$ as given in Eq. (19). Thus it can be seen that only cuckoos with good (i.e. low) fitness values and far from $\boldsymbol{z}_{\mathrm{FT}}$ are assigned with big $FS(\boldsymbol{z}_i)$. By contrast, for cuckoos having bad (i.e. high) fitness values or near $\boldsymbol{z}_{\mathrm{FT}}$, the small values of $FS(\boldsymbol{z}_i)$ are obtained as follows:

$$FS(\boldsymbol{z}_i) = W(\boldsymbol{z}_i) \cdot \sqrt{\sum_{d=1}^{D} (\boldsymbol{z}_{i,d} - \boldsymbol{z}_{\mathrm{FT},d})^2} \tag{19}$$

Finally $\boldsymbol{z}_{\mathrm{FST}}$ is identified from the $FS(\boldsymbol{z}_i)$ of all cuckoos and $\boldsymbol{z}_i$ with the biggest $FS$ is assigned to $\boldsymbol{z}_{\mathrm{FST}}$. Mathematically, $\boldsymbol{z}_{\mathrm{FST}}$ is identified as follows:

$$\boldsymbol{z}_{\mathrm{FST}} = \arg\max_{\forall i \in [1, S]} (FS(\boldsymbol{z}_i)) \tag{20}$$

As is clear from the above descriptions, the introduction of $\boldsymbol{z}_{\mathrm{FST}}$ has a possibility to guide the search to a potentially more promising region that is far from the current global optimum.

In general, there are three modes to perform the BRL-based search strategy, that is, (1) to perform it only at the first component of MLCS, (2) to perform it only at the second component of MLCS, and (3) to perform it at both components. From the experiments, we find that different modes yield different results on the same problem, and different modes are needed to yield the best performance for different problems. To resolve the problem in a relatively general manner, we propose to set the performing mode of the BRL-based search strategy such that one of the three modes is chosen randomly for each iteration.

*3.2.3. Complete framework of MLCS*

Here we integrate the aforementioned search strategies and present the complete implementation of MLCS as illustrated in Fig. 4.

**Algorithm 1.** Implementation of MLCS.

Set the population size $S$, maximum iteration number $g_{max}$, discovering probability $p_a$, control coefficient $c_1$, learning coefficient $c_2$, and search space dimension $D$;
Initialize a population of $S$ solutions $\boldsymbol{z}_i$ ($i = 1, 2, \cdots, S$);
Calculate the fitness value of each solution $f_i$ ($i = 1, 2, \cdots, S$);
Identify $\boldsymbol{z}_B$, $\boldsymbol{z}_{FT}$ and $\boldsymbol{z}_{FST}$ from the current $S$ solutions;
$g = 1$;
**While** $g \leq g_{max}$ **do**
    Randomly select an integer from 1 to 3 and assign it to a flag variable *flag*;
    Select an exemplar ($\boldsymbol{z}_{ei}$) for each cuckoo $i$, and assign its series number to $e_i$;
    **for** $i = 1$ to $S$ **do**
        **if** (*flag* == 1 OR *flag* == 3) AND $\boldsymbol{z}_i = \boldsymbol{z}_B$ **then**
            Generate a new solution $\boldsymbol{z}_i^{new}$ using Eq. (16);
        **else**
            Generate a new solution $\boldsymbol{z}_i^{new}$ using Eq. (10);
        **end if**
        Calculate its fitness value $f_i^{new}$;
        **if** $f_i^{new} < f_i$ **then**
            $\boldsymbol{z}_i = \boldsymbol{z}_i^{new}$ and $f_i = f_i^{new}$;
        **end if**
    **end for**
    Identify $\boldsymbol{z}_B$, $\boldsymbol{z}_{FT}$ and $\boldsymbol{z}_{FST}$ from the current $S$ solutions;
    **for** $i = 1$ to $S$ **do**
        **if** (*flag* == 2 OR *flag* == 3) AND $\boldsymbol{z}_i = \boldsymbol{z}_B$ **then**
            Generate a new solution $\boldsymbol{z}_i^{new}$ using Eq. (16);
        **else**
            Generate a new solution $\boldsymbol{z}_i^{new}$ using Eq. (11);
        **end if**
        Repeat steps 15-18 to update $\boldsymbol{z}_i$ and $f_i$;
    **end for**
    Identify $\boldsymbol{z}_B$, $\boldsymbol{z}_{FT}$ and $\boldsymbol{z}_{FST}$ from the current $S$ solutions;
    $g = g + 1$;
**end while**

One Iterative Optimization Operation

**Fig. 4.** Complete framework of MLCS.

### *3.2.4. Complexity analysis of MLCS*

For ease of analysis, firstly the time complexity of the basic CS algorithm is given in this subsection. For each cuckoo, we need to perform $O$ ($D$) number of operations for a single iteration, resulting in $O$ ($S$. $D$) complexity. However, generally CS runs for a number of iterations, so the overall complexity depends on the maximum iteration number ($g_{\max}$). This procedure gives the overall time complexity of the basic CS as $O$ ($S$. $D$. $g_{\max}$). Compared with the basic CS, MLCS needs to perform additional computations of $O$ ($S$. $D$. $g_{\max}$) on the ML-based and BRL-based evolutionary equations. Meanwhile, the selection of exemplars and determination of the two top cuckoos consume further computational complexity of $O$ ($S$. $D$. $g_{\max}$). In accordance with these observations above, the time complexity of MLCS is the same as that of the basic CS algorithm, i.e. $O$ ($S$. $D$. $g_{\max}$). However, MLCS significantly outperforms the basic CS in terms of search accuracy, search efficiency (i.e. convergence speed), and search reliability according to the experimental results given in the following sections. These observations suggest that our proposed MLCS achieves better tradeoff between performance improvement and time complexity compared with the basic CS.

### *3.3. Validation of MLCS with benchmark functions*

#### *3.3.1. Benchmark functions and performance metrics*

To verify the search capability of MLCS, it is compared with six CS-based algorithms on 20 scalable benchmark functions [37, 45, 46] listed in Table 1. These benchmark functions can be divided into two categories: unimodal problems and multimodal problems. Hereinto, $f_1$-$f_5$ are unimodal ones each containing only one optimum. The exploitation capability of optimization algorithms is commonly tested on this type of benchmark functions. Whereas $f_6$-$f_{20}$ are multimodal ones with each having numerous local optima but only one global optimum. Besides the exploitation capability, the exploration capability of optimization algorithms is commonly tested on multimodal benchmark functions. Table 1 gives the search range, function value at the

global optimum ($F_m$) and acceptable value ($\varepsilon$) of each benchmark function. More specially, when the lowest fitness value (in this study, the fitness value of each benchmark function is defined as the difference between the function value found by an algorithm and $F_m$) obtained by an algorithm is less than $\varepsilon$, the run is regarded as a successful run.

**Table 1** Benchmark functions used.

| Function name | Mathematical formula | Range | $F_m$ | $\varepsilon$ |
|---|---|---|---|---|
| Sphere function | $f_1(\boldsymbol{x}) = \sum_{d=1}^{D} x_d^2$ | $[-100, 100]^D$ | 0 | 1.0e-6 |
| Schwefel's problem 2.22 | $f_2(\boldsymbol{x}) = \sum_{d=1}^{D} \lvert x_d \rvert + \prod_{d=1}^{D} \lvert x_d \rvert$ | $[-10, 10]^D$ | 0 | 1.0e-6 |
| High conditioned elliptic function | $f_3(\boldsymbol{x}) = \sum_{d=1}^{D} (10^6)^{\frac{d-1}{D-1}} x_d^2$ | $[-100, 100]^D$ | 0 | 1.0e-6 |
| Schwefel's problem 2.21 | $f_4(\boldsymbol{x}) = \max_d \{\lvert x_d \rvert,\ 1 \le d \le D\}$ | $[-100, 100]^D$ | 0 | 1.0e-6 |
| Sum square function | $f_5(\boldsymbol{x}) = \sum_{d=1}^{D} d x_d^2$ | $[-10, 10]^D$ | 0 | 1.0e-6 |
| Griewank's function | $f_6(\boldsymbol{x}) = \sum_{d=1}^{D} \frac{x_d^2}{4000} - \prod_{d=1}^{D} \cos(\frac{x_d}{\sqrt{d}}) + 1$ | $[-600, 600]^D$ | 0 | 1.0e-2 |
| Rastrigin's function | $f_7(\boldsymbol{x}) = \sum_{d=1}^{D} (x_d^2 - 10\cos(2\pi x_d) + 10)$ | $[-5.12, 5.12]^D$ | 0 | 1.0e-2 |
| Noncontinuous Rastrigin's function | $f_8(\boldsymbol{x}) = \sum_{d=1}^{D} (y_d^2 - 10\cos(2\pi y_d) + 10),\ y_d = \begin{cases} x_d & \lvert x_d \rvert < \frac{1}{2} \\ \frac{round(2x_d)}{2} & \lvert x_d \rvert \ge \frac{1}{2} \end{cases}$ | $[-5.12, 5.12]^D$ | 0 | 1.0e-2 |
| Ackley's function | $f_9(\boldsymbol{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} x_d^2}) - \exp(\frac{1}{D}\sum_{d=1}^{D} \cos(2\pi x_d)) + 20 + e$ | $[-32.768, 32.768]^D$ | 0 | 1.0e-2 |
| Schwefel's function | $f_{10}(\boldsymbol{x}) = 418.98288727243369 \cdot D - \sum_{d=1}^{D} x_d \sin(\lvert x_d \rvert^{\frac{1}{2}})$ | $[-500, 500]^D$ | 0 | 1.0e-2 |
| Weierstrass function | $f_{11}(\boldsymbol{x}) = \sum_{d=1}^{D} (\sum_{k=0}^{k_{\max}} a^k \cos(2\pi b^k (x_d + 0.5))) - D\sum_{k=0}^{k_{\max}} a^k \cos(2\pi b^k \cdot 0.5)$ where $a = 0.5, b = 3, k_{\max} = 20$ | $[-0.5, 0.5]^D$ | 0 | 1.0e-2 |
| Penalized function1 | $f_{12}(\boldsymbol{x}) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{d=1}^{D-1} (y_d - 1)^2 \cdot [1 + 10\sin^2(\pi y_{d+1})] + (y_D - 1)^2\right\} + \sum_{d=1}^{D} u(x_d, 10, 100, 4)$ where $y_d = 1 + \frac{1}{4}(x_d + 1),\ u(x_d, a, k, m) = \begin{cases} k(x_d - a)^m & x_d > a \\ 0 & -a \le x_d \le a \\ k(-x_d - a)^m & x_d < -a \end{cases}$ | $[-50, 50]^D$ | 0 | 1.0e-2 |
| Penalized function2 | $f_{13}(\boldsymbol{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{d=1}^{D-1} (x_d - 1)^2 \cdot [1 + \sin^2(\pi x_{d+1})] + \cdots \cdots (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\} + \sum_{d=1}^{D} u(x_d, 5, 100, 4)$ | $[-50, 50]^D$ | 0 | 1.0e-2 |
| Schaffer function | $f_{14}(\boldsymbol{x}) = 0.5 + (\sin^2(\sqrt{\sum_{d=1}^{D} x_d^2}) - 0.5) \Big/ (1 + 0.001(\sum_{d=1}^{D} x_d^2))^2$ | $[-100, 100]^D$ | 0 | 1.0e-2 |
| Alpine function | $f_{15}(\boldsymbol{x}) = \sum_{d=1}^{D} \lvert x_d \cdot \sin(x_d) + 0.1 \cdot x_d \rvert$ | $[-10, 10]^D$ | 0 | 1.0e-2 |
| Levy function | $f_{16}(\boldsymbol{x}) = \sum_{d=1}^{D-1} (x_d - 1)^2 \cdot [1 + \sin^2(3\pi x_{d+1})] + \sin^2(3\pi x_1) + \lvert x_D - 1 \rvert [1 + \sin^2(3\pi x_D)]$ | $[-10, 10]^D$ | 0 | 1.0e-2 |

**Table 1** (*continued*).

| Function name | Mathematical formula | Range | $F_m$ | $\varepsilon$ |
|---|---|---|---|---|
| Shifted Rastrigin | $f_{17}(\boldsymbol{x}) = f_7(\boldsymbol{x}-\boldsymbol{o}) + f_{bias1}, \quad f_{bias1} = -330$ | $[-5.12, 5.12]^D$ | -330 | 1.0e-2 |
| Shifted Noncontinuous Rastrigin | $f_{18}(\boldsymbol{x}) = f_8(\boldsymbol{x}-\boldsymbol{o}) + f_{bias2}, \quad f_{bias2} = -330$ | $[-5.12, 5.12]^D$ | -330 | 1.0e-2 |
| Shifted Griewank | $f_{19}(\boldsymbol{x}) = f_6(\boldsymbol{x}-\boldsymbol{o}) + f_{bias3}, \quad f_{bias3} = -180$ | $[-600, 600]^D$ | -180 | 1.0e-2 |
| Shifted Ackley | $f_{20}(\boldsymbol{x}) = f_9(\boldsymbol{x}-\boldsymbol{o}) + f_{bias4}, \quad f_{bias4} = -140$ | $[-32.768, 32.768]^D$ | -140 | 1.0e-2 |

To evaluate the performance of involved optimization algorithms, two commonly-used and important criteria, the search accuracy and the search efficiency (i.e. convergence speed), are utilized, and here they are measured via the mean lowest fitness ($F_{\text{mean}}$) and success performance (*SP*) [47], respectively. $F_{\text{mean}}$ is defined as the mean of the lowest fitness values obtained by the algorithm. Smaller $F_{\text{mean}}$ value implies that the algorithm has better search accuracy. The convergence speed of an algorithm in attaining the solution with the acceptable value $\varepsilon$ is measured by *SP*, which is computed as

$$SP = \frac{\#MI \cdot \#TRs}{\#SRs} \tag{21}$$

where *#MI* is the mean iteration number needed to reach $\varepsilon$ (it should be pointed out that *#MI* will only be calculated for successful runs); *#TRs* is the number of total runs; and *#SRs* is the number of successful runs. Obviously, smaller *SP* implies that the algorithm requires less cost to solve the problem. In addition, Wilcoxon rank test is also performed for the purpose of further validating the significance of performance differences between the proposed MLCS algorithm and other CS-based algorithms. This test is a non-parametric statistical procedure [48, 49] which is employed to perform pairwise comparison between MLCS and other CS-based algorithms. The outputs of Wilcoxon rank test are shown by the *h* values, where the *h* values of "–", "=", and "+" denote the competitor performs significantly worse than, insignificantly different from, and significantly better than MLCS, respectively. Apart from these, the convergence curve of each

benchmark function is also illustrated to enable us to compare the search accuracy and convergence speed of all involved CS-based algorithms qualitatively.

### *3.3.2. Comparison of MLCS with other CS-based algorithms*

To show the competitiveness of MLCS, in this subsection we compare it with the basic CS and five new CS variants, namely CCS [37], ACS [38], ICS [39], NNCS [40], and NoCuSa [41] based on the 20 benchmark functions with dimension equal to 50 (i.e. $D$ = 50). Parameter settings of the five new CS variants used in the comparison are set according to their original references. For MLCS and the basic CS, the value of their common parameter $p_a$ is set according to recommendation of Yang and Deb [18, 42], that is, $p_a$ = 0.25. The specific parameters of MLCS, namely $c_1$ and $c_2$ are adjusted by experiment analysis, from which it was observed that $c_1$ = 0.09 and $c_2$ = 1.25 are capable of balancing the search performance of MLCS on a diverse range of problems. For the sake of fair comparison, the population size $S$ and the maximum iteration number $g_{\max}$ used to terminate the algorithms are uniformly set to $D$ and 5000 respectively, as similar done in some existing literatures [40, 50]. All CS-based algorithms involved are run 30 times independently for each benchmark function to reduce random discrepancy. For clarity, the parameter settings of all involved algorithms are summarized in Table 2, where CS denotes the basic CS.

**Table 2** Parameter settings of the involved CS-based algorithms.

| Algorithm | Parameter settings |
|---|---|
| CS | $S = D$, $p_a = 0.25$, $\alpha = 0.01$, $g_{\max} = 5000$ |
| CCS | $S = D$, $p_a = 0.25$, $g_{\max} = 5000$, generation of chaotic sequences: Gauss map |
| ACS | $S = D$, $p_a = 0.25$, $g_{\max} = 5000$ |
| ICS | $S = D$, $p_{a\max} = 0.5$, $p_{a\min} = 0.005$, $\alpha_{\max} = 0.5$, $\alpha_{\min} = 0.01$, $g_{\max} = 5000$ |
| NNCS | $S = D$, $p_a = 0.25$, $p = 0.25$, $g_{\max} = 5000$, selection of nearest neighbour solutions: fitness-based similar metrics |
| NoCuSa | $S = D$, $p_a = 0.3$, $\alpha = 1.1$, $\beta = 1.7$, $\delta = 1.6$, $g_{\max} = 5000$ |
| MLCS | $S = D$, $p_a = 0.25$, $\alpha = 0.01$, $c_1 = 0.09$, $c_2 = 1.25$, $g_{\max} = 5000$ |

Table 3 reveals the results of $F_{\text{mean}}$, *SD* (the standard deviation of lowest fitness values achieved by an algorithm), *h* (at a 0.05 significance level), and *SP* produced by involved algorithms. For clarity, the best results on each benchmark function are marked in boldface. The comparison results of $F_{\text{mean}}$ between MLCS and other CS-based algorithms are summed up as *#BMF*, which is the number of the best (namely lowest) $F_{\text{mean}}$ obtained with each CS-based algorithm. The *h* result is summed up as "–/=/+" to represent the number of functions on which MLCS performs significantly better than, insignificantly different from, and significantly worse than its competitor, respectively. The results of *SP* are summed up as *#FSP* to denote the number of fastest (namely lowest) *SP* obtained by each involved CS-based algorithm. Fig. 5 presents the evolution progress in terms of the $F_{\text{mean}}$ of each algorithm for each benchmark function in 30 independent runs.

**Table 3** $F_{mean}$, *SD*, *h*, and *SP* values produced by the seven CS-based algorithms on 20 benchmark functions.

| | | CS | CCS | ACS | ICS | NNCS | NoCuSa | MLCS |
|---|---|---|---|---|---|---|---|---|
| $f_1$ | $F_{mean}$ | 1.0917e-16 | 4.8888e-27 | 7.6289e-15 | 9.1140e-27 | 5.3304e-26 | 1.4338e-87 | **4.0307e-115** |
| | *SD* | 3.5365e-17 | 2.6715e-27 | 2.4700e-14 | 3.2638e-27 | 2.1116e-26 | 3.6664e-87 | **7.7653e-115** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 2.6178e+03 | 1.6888e+03 | 2.5021e+03 | 1.7254e+03 | 1.7761e+03 | 5.3850e+02 | **4.5447e+02** |
| $f_2$ | $F_{mean}$ | 1.0270e-07 | 3.4964e-17 | 1.2675e-07 | 4.6743e-15 | 2.1274e-15 | 6.8531e-48 | **3.1990e-67** |
| | *SD* | 2.1825e-08 | 1.2586e-17 | 1.2831e-07 | 1.3085e-15 | 4.8792e-16 | 2.2416e-47 | **2.2407e-67** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 4.4834e+03 | 2.1742e+03 | 4.3974e+03 | 2256 | 2.4188e+03 | 7.6087e+02 | **6.0850e+02** |
| $f_3$ | $F_{mean}$ | 2.0627e-13 | 2.3308e-24 | 7.3345e-11 | 3.5250e-24 | 4.0667e-23 | 6.8653e-84 | **9.0651e-108** |
| | *SD* | 4.9563e-14 | 1.5549e-24 | 1.8685e-10 | 7.8858e-25 | 1.1728e-23 | 1.6825e-83 | **1.0249e-107** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 3397 | 2.1332e+03 | 3.5814e+03 | 2.0583e+03 | 2.2679e+03 | 7.3867e+02 | **6.2663e+02** |
| $f_4$ | $F_{mean}$ | 5.5877e-02 | 2.1050e-01 | 1.2714e-01 | 2.4550e-02 | 3.7179e-02 | 4.0351e+00 | **3.7238e-03** |
| | *SD* | 5.5453e-03 | 2.2243e-02 | 2.8616e-02 | **2.0881e-03** | 5.0812e-03 | 1.5168e+00 | 9.0304e-03 |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **143430** |
| $f_5$ | $F_{mean}$ | 2.4498e-17 | 8.1016e-28 | 1.2375e-15 | 1.8718e-27 | 9.8225e-27 | 1.8248e-88 | **1.7775e-112** |
| | *SD* | 7.5495e-18 | 3.4048e-28 | 4.6804e-15 | 6.7522e-28 | 4.7647e-27 | 4.0412e-88 | **2.4550e-112** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 2.4535e+03 | 1.5685e+03 | 2.2504e+03 | 1.6306e+03 | 1.6573e+03 | 4.9237e+02 | **4.3790e+02** |
| $f_6$ | $F_{mean}$ | 3.9779e-11 | **0** | 2.8771e-09 | **0** | **0** | 8.4212e-03 | **0** |
| | *SD* | 9.6550e-11 | **0** | 5.8763e-09 | **0** | **0** | 1.6462e-02 | **0** |
| | *h* | – | = | – | = | = | – | |
| | *SP* | 1.7856e+03 | 1.1570e+03 | 1.5609e+03 | 1.2844e+03 | 1.1921e+03 | 4.5161e+02 | **3.1973e+02** |
| $f_7$ | $F_{mean}$ | 9.1815e+01 | 1.0852e+02 | 1.3818e+02 | 4.2431e+01 | 6.2051e+01 | 3.5723e+01 | **0** |
| | *SD* | 6.1838e+00 | 8.2038e+00 | 1.4751e+01 | 4.4917e+00 | 9.2665e+00 | 2.1744e+01 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **1.4567e+03** |
| $f_8$ | $F_{mean}$ | 8.0107e+01 | 9.3356e+01 | 1.2381e+02 | 4.1932e+01 | 4.9470e+01 | 4.6002e+01 | **0** |
| | *SD* | 7.6650e+00 | 9.7529e+00 | 1.2016e+01 | 3.2452e+00 | 6.1738e+00 | 1.7954e+01 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **2.3606e+03** |
| $f_9$ | $F_{mean}$ | 3.2648e-03 | 4.8672e-14 | 5.0895e-06 | 1.1866e-13 | 1.9102e-13 | 1.3895e+00 | **6.1580e-15** |
| | *SD* | 2.7273e-03 | 1.1645e-14 | 1.7878e-06 | 1.4950e-14 | 4.8736e-14 | 7.1662e-01 | **1.5711e-15** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 4.8057e+03 | 1.2841e+03 | 2.5752e+03 | 1.4120e+03 | 1.4098e+03 | 2.3028e+03 | **3.2397e+02** |
| $f_{10}$ | $F_{mean}$ | 4.7823e+03 | 5.5729e+03 | 5.5705e+03 | 3.5621e+03 | 1.3844e+03 | 7.2062e+03 | **7.8959e+00** |
| | *SD* | 3.1615e+02 | 2.9743e+02 | 4.2298e+02 | 3.0613e+02 | 5.2420e+02 | 1.2509e+03 | **2.9544e+01** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **1.9006e+03** |
| $f_{11}$ | $F_{mean}$ | 7.4593e-01 | 5.8776e-12 | 5.9253e-03 | 5.7090e-05 | 3.4750e-12 | 2.3211e+00 | **0** |
| | *SD* | 1.0140e-01 | 7.8980e-12 | 2.2663e-03 | 2.0661e-05 | 1.8803e-12 | 1.1019e+00 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | 2.5275e+03 | 4.8416e+03 | 2.7059e+03 | 2.4761e+03 | Null | **8.1770e+02** |

**Table 3** (*continued*).

| | | CS | CCS | ACS | ICS | NNCS | NoCuSa | MLCS |
|---|---|---|---|---|---|---|---|---|
| $f_{12}$ | $F_{mean}$ | 2.8124e+01 | 2.0054e-01 | 2.4683e+01 | 1.2382e+00 | 2.0263e-12 | 3.5245e-02 | **0** |
| | *SD* | 5.5402e+00 | 5.2367e-01 | 7.1158e+00 | 7.3822e-01 | 7.5745e-12 | 5.4889e-02 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | 6.4078e+03 | Null | Null | 2.7781e+03 | 5.5481e+02 | **3.9360e+02** |
| $f_{13}$ | $F_{mean}$ | 6.8539e-11 | 3.0012e-24 | 1.2920e-09 | 3.1886e-22 | 1.7037e-24 | 9.2033e-02 | **0** |
| | *SD* | 4.4048e-11 | 2.1226e-24 | 1.6579e-09 | 2.3385e-22 | 1.0464e-24 | 4.4684e-01 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 2.9298e+03 | 1.4862e+03 | 2.5669e+03 | 1.7169e+03 | 1.3424e+03 | 8.8479e+02 | **3.5393e+02** |
| $f_{14}$ | $F_{mean}$ | 3.3583e-01 | 1.5109e-01 | 3.1581e-01 | 2.3505e-01 | 2.0131e-01 | 1.7799e-01 | **1.0438e-01** |
| | *SD* | 2.3369e-02 | 2.5196e-02 | 2.2116e-02 | **2.0816e-02** | 2.4678e-02 | 3.4663e-02 | 3.0559e-02 |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | Null |
| $f_{15}$ | $F_{mean}$ | 1.1428e+01 | 5.4766e+00 | 1.2937e+01 | 1.6122e+00 | 3.0164e+00 | **3.5724e-16** | 3.9281e-05 |
| | *SD* | 1.1562e+00 | 7.5880e-01 | 1.6225e+00 | 4.3449e-01 | 9.5087e-01 | **4.4954e-16** | 3.2744e-05 |
| | *h* | – | – | – | – | – | + | |
| | *SP* | Null | Null | Null | Null | Null | **2.9877e+02** | 6.9720e+02 |
| $f_{16}$ | $F_{mean}$ | 8.1879e-15 | 1.1087e-27 | 4.9959e-12 | 5.6483e-27 | 1.1959e-26 | 3.1182e-01 | **0** |
| | *SD* | 3.5029e-15 | 9.7784e-28 | 7.7812e-12 | 4.3291e-27 | 7.0787e-27 | 4.7743e-01 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 1.8781e+03 | 9.6317e+02 | 1.6176e+03 | 1.1237e+03 | 9.7173e+02 | 6.6604e+02 | **2.3210e+02** |
| $f_{17}$ | *Mean* | 7.7573e+01 | 1.2532e+02 | 1.2505e+02 | 4.0058e+01 | 5.4009e+01 | 1.3677e+02 | **6.3014e-01** |
| | *SD* | 7.2116e+00 | 9.8449e+00 | 1.0686e+01 | 2.9734e+00 | 5.9123e+00 | 3.0449e+01 | **7.9112e-01** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **2.9452e+03** |
| $f_{18}$ | *Mean* | 8.0345e+01 | 1.1727e+02 | 1.0714e+02 | 3.7352e+01 | 4.4932e+01 | 1.4463e+02 | **1.4667e+00** |
| | *SD* | 9.7168e+00 | 1.0542e+01 | 8.8728e+00 | 3.8051e+00 | 4.9992e+00 | 3.6090e+01 | **1.1175e+00** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | Null | Null | Null | Null | Null | **1.6153e+04** |
| $f_{19}$ | *Mean* | 1.7195e-11 | 2.8422e-14 | 1.1393e-08 | 2.8422e-14 | 2.8422e-14 | 9.9236e-03 | **0** |
| | *SD* | 1.8534e-11 | **0** | 1.9747e-08 | 1.3024e-14 | 8.5265e-15 | 1.3214e-02 | **0** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | 1.8176e+03 | 1.2539e+03 | 1.6450e+03 | 1.2573e+03 | 1.1879e+03 | 6.0050e+02 | **3.2373e+02** |
| $f_{20}$ | *Mean* | 1.9427e+00 | 3.4106e-13 | 7.6652e-02 | 6.6677e-11 | 1.3642e-12 | 1.4390e+00 | **2.8422e-14** |
| | *SD* | 3.2558e-01 | 9.2048e-14 | 1.9273e-01 | 1.4971e-10 | 9.0137e-13 | 6.7924e-01 | **1.2569e-14** |
| | *h* | – | – | – | – | – | – | |
| | *SP* | Null | 1.4350e+03 | 4.3573e+04 | 1.6416e+03 | 1.5515e+03 | 3.8400e+03 | **3.4327e+02** |
| *#BMF* | | 0 | 1 | 0 | 1 | 1 | 1 | **19** |
| –/=/+ | | 20/0/0 | 19/1/0 | 20/0/0 | 19/1/0 | 19/1/0 | 19/0/1 | |
| *#FSP* | | 0 | 0 | 0 | 0 | 0 | 1 | **18** |

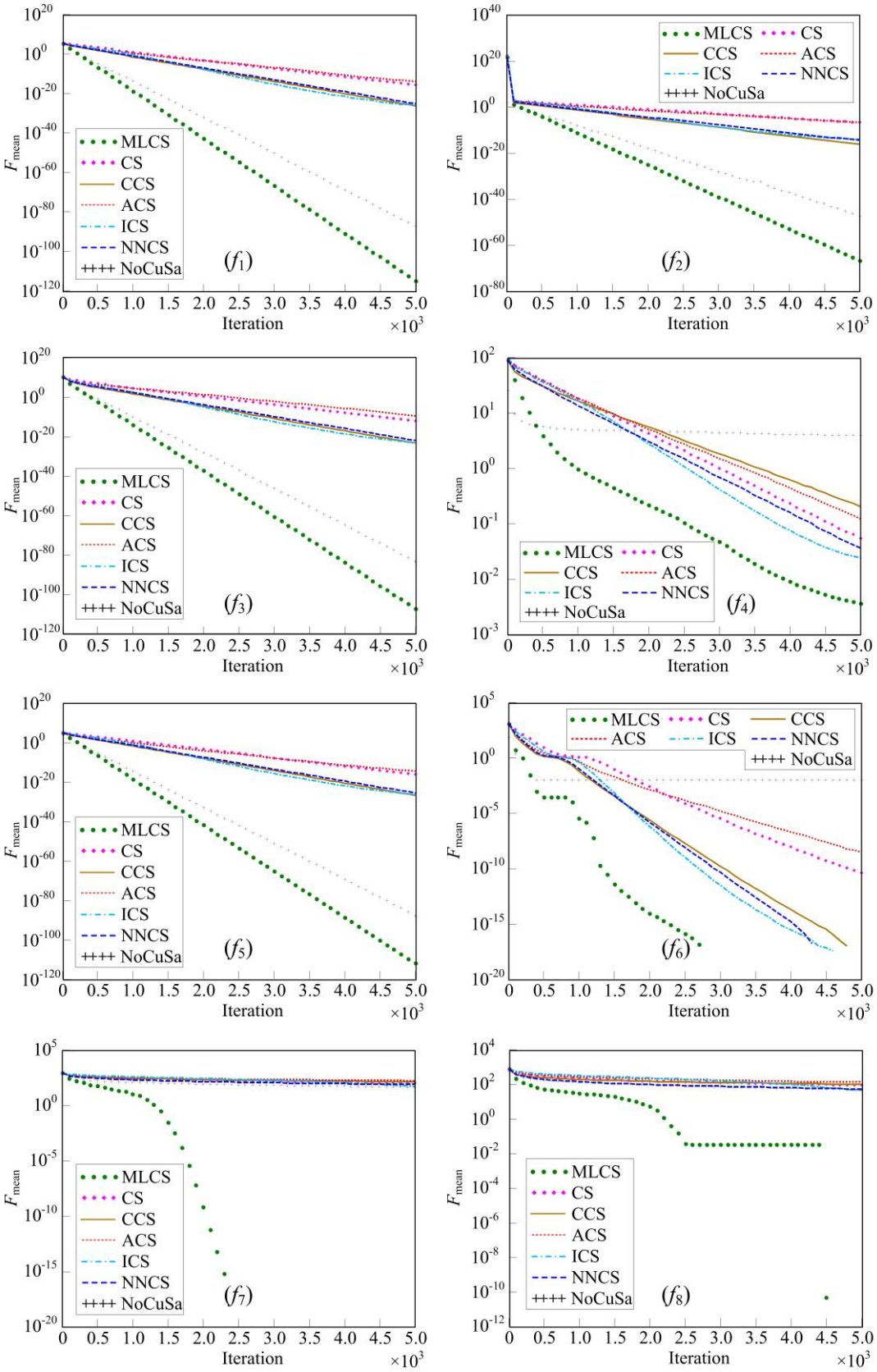


**Fig. 5.** Mean convergence characteristics of seven CS-based algorithms on 20 benchmark functions.

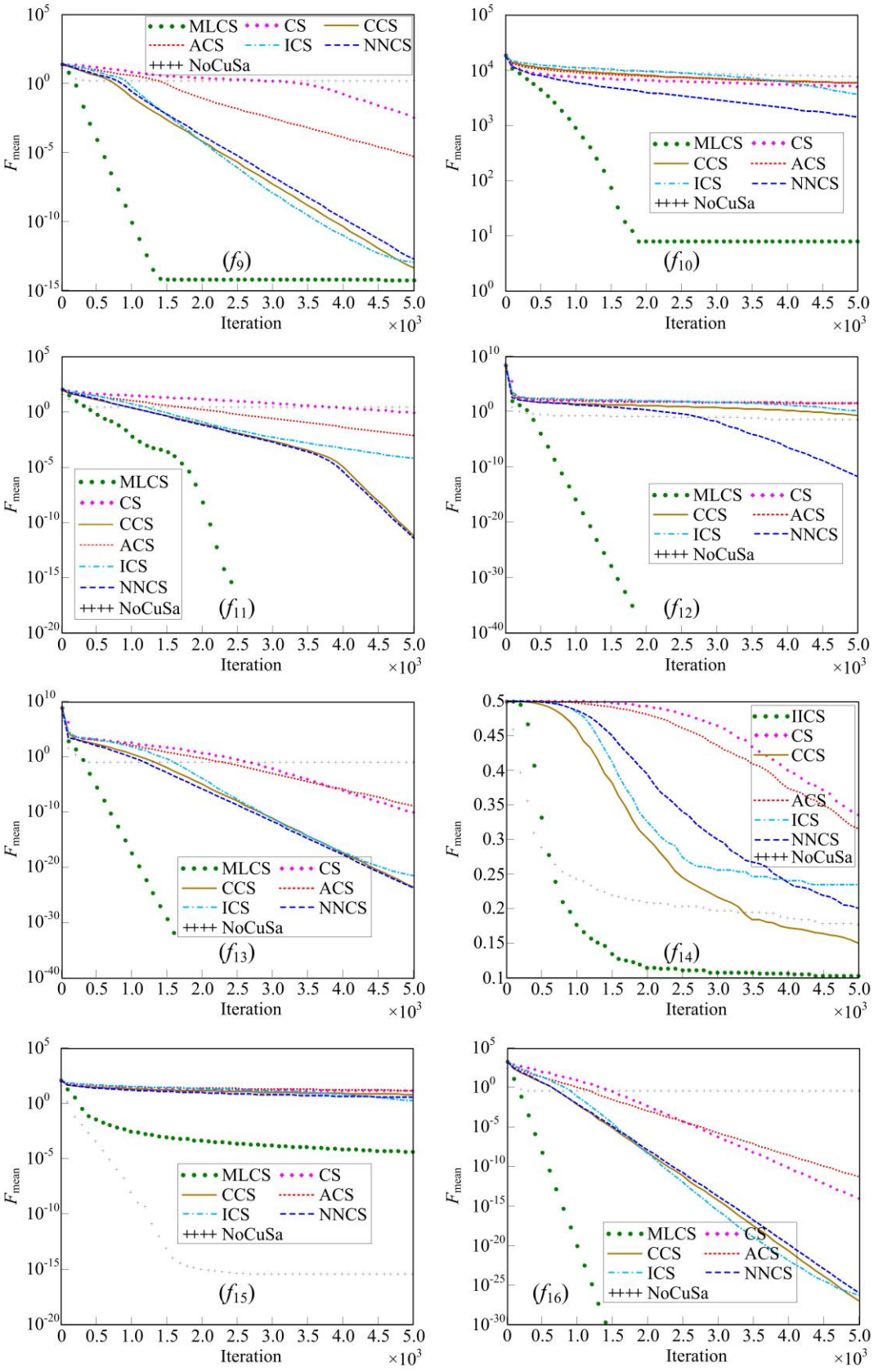


**Fig. 5.** (*continued*).

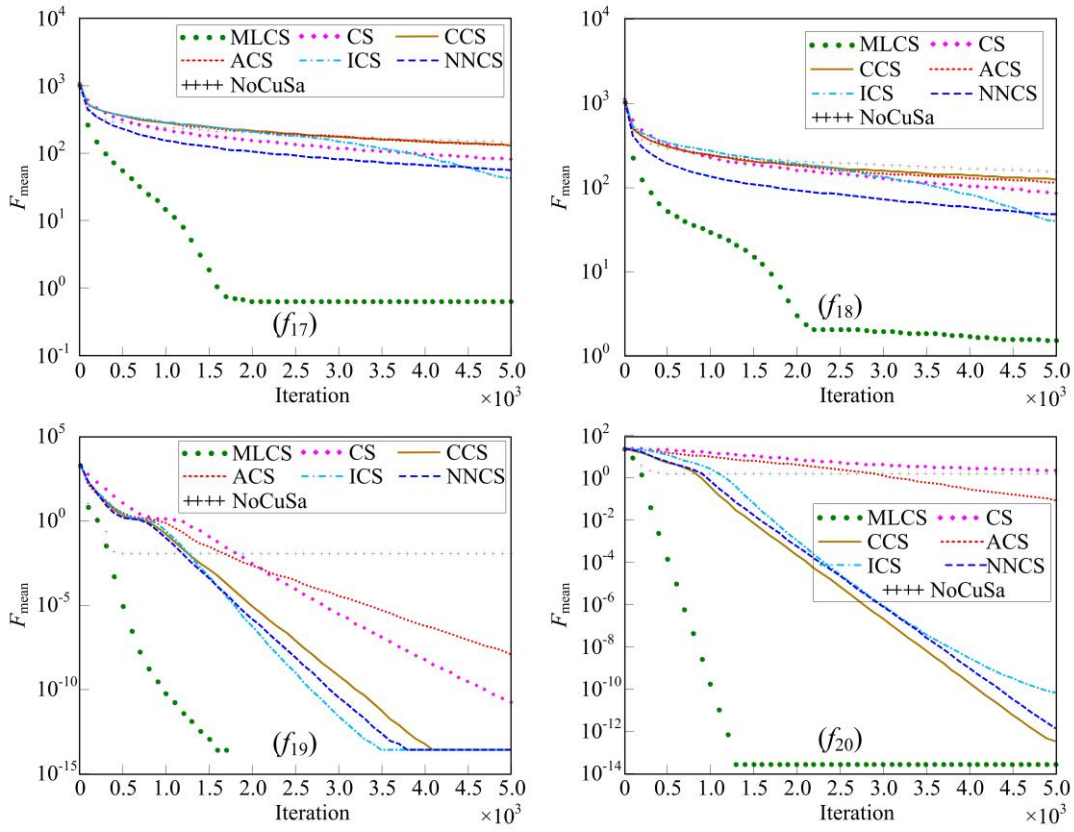


**Fig. 5.** (*continued*).

From the $F_{mean}$, *SD*, and *h* values in Table 3, along with the convergence curves in Fig. 5, we can find that MLCS has the most excellent search accuracy. To be specific, the *#BMF* value achieved by MLCS is 19 on the 20 benchmark functions. In terms of $F_{mean}$, MLCS surpasses the other six CS-based algorithms on 18 of the 20 benchmark functions (i.e. all functions except $f_6$ and $f_{15}$), and particularly remarkably enhances the search accuracy on $f_7$, $f_8$, $f_{10}$, and $f_{17}$. The benchmark function $f_6$ seems relatively easier to be optimized since MLCS, CCS, ICS, and NNCS successfully locate its global optimum in all their 30 independent runs. As for $f_{15}$, it can be found that only MLCS and NoCuSa are capable of locating the near global optima of this function, and MLCS attains the second best $F_{mean}$ value. Moreover, MLCS successfully finds the global optima of 8 of the 15 multimode benchmark functions in all 30 runs, while other algorithms do only once on the relatively simple Griewank's function (i.e. $f_6$). Also, it is worthy to point out that our

MLCS is the sole one which is capable of finding the near global optima of $f_{10}$ with a minimal accuracy level of $10^{-11}$ in 28 of 30 independent runs, while the best accuracy level achieved by other algorithms is $10^{2}$. The values of *h* listed in Table 3 reveal the pairwise comparison results between MLCS and other CS-based algorithms on each employed benchmark function. It can be seen that the values of *h* obtained from Wilcoxon rank test are in accordance with the values of $F_{\text{mean}}$ since the number of benchmark functions on which MLCS remarkably surpasses the other CS-based algorithms is bigger than the number of benchmark functions on which MLCS remarkably underperforms the other algorithms. In fact, the ratio of the former number and the latter number is 19:1. Also, from Table 3 it can be observed that overall MLCS has smaller *SD* values in comparison with other CS-based algorithms, which exhibits its superior reliability over its competitors.

The *SP* values in Table 3 and the convergence curves in Fig. 5 demonstrate the algorithms' search efficiency on the 20 benchmark functions quantitatively and qualitatively respectively. It should be noted here that the *SP* value represents the computation cost by the algorithm to locate any solution whose fitness value is less than $\varepsilon$, so it would no way to get the value of *SP* if the algorithm never achieves $\varepsilon$ within $g_{\max}$. In this case, the value of *SP* will be denoted by "Null", and only the convergence curve shown in Fig. 5 can be applied for justifying the search efficiency of the algorithm. From the *SP* analysis listed in Table 3, it can be seen that MLCS has the most competitive search efficiency among the compared CS-based algorithms because MLCS produces 18 fastest *SP* values on $f_1$-$f_{13}$ and $f_{16}$-$f_{20}$. Also, the excellent search efficiency of MLCS on these benchmark functions can be supported by the corresponding convergence curves as given in Fig. 5. Moreover, it can be found from Fig. 5 that MLCS converges faster than any other algorithms on $f_{14}$, besides the above 18 benchmark functions, and for $f_{15}$, MLCS achieves the second fastest convergence speed only next to NoCuSa.

On the basis of the analyses above, it can be concluded that compared with the competitors MLCS has overwhelming superiority in all aspects having been utilized for comparison, i.e. the search accuracy, the search efficiency, and the search reliability. Such a promising search performance offers MLCS a greater chance to produce better results for complicated optimization tasks than the competitors. Therefore, we consider the use of MLCS for developing the proposed FW-SLFN model in the following sections.

## 4. MLCS optimized FW-SLFN model

MLCS is used for the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ in the prediction model described by Eq. (2). In the search process, each cuckoo $i$, encoded as $\boldsymbol{z}_i = [\boldsymbol{w}_i, \boldsymbol{P}_{\mathrm{N}i}]$, denotes a set of candidate combination values of $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$, and Eq. (3) is utilized as the fitness function to evaluate the quality of each cuckoo. The procedures for calculating cuckoo $i$'s fitness value $J_i$, are given by Algorithm 2 shown in Fig. 6.

**Algorithm 2.** Calculation for $J_i$.

**Input:** $\boldsymbol{z}_i = [\boldsymbol{w}_i, \boldsymbol{P}_{\mathrm{N}i}]$.

**Output:** $J_i$.

1: Substitute the values of the parameter vectors $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ corresponding to cuckoo $i$ (i.e., $\boldsymbol{w}_i$ and $\boldsymbol{P}_{\mathrm{N}i}$) into the prediction model described by Eq. (2);

2: **for** $m = 1$ to $M$ **do**

3: Based on the normalized data of the $m$th feature vector $\boldsymbol{frt}_m$, obtain the corresponding predicted value of MSET, i.e., $\hat{\boldsymbol{x}}_m$ ;

4: **end for**

5: $J_i = \sum_{m=1}^{M} \left| \frac{x_m - \hat{x}_m}{x_m} \right| \Big/ M.$

**Fig. 6.** Procedures for calculating the fitness value of cuckoo $i$.

The implementation of MLCS for the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ is shown in Fig. 7, where the iteration terminates when either $g > g_{\max}$ or the early stopping condition is satisfied.

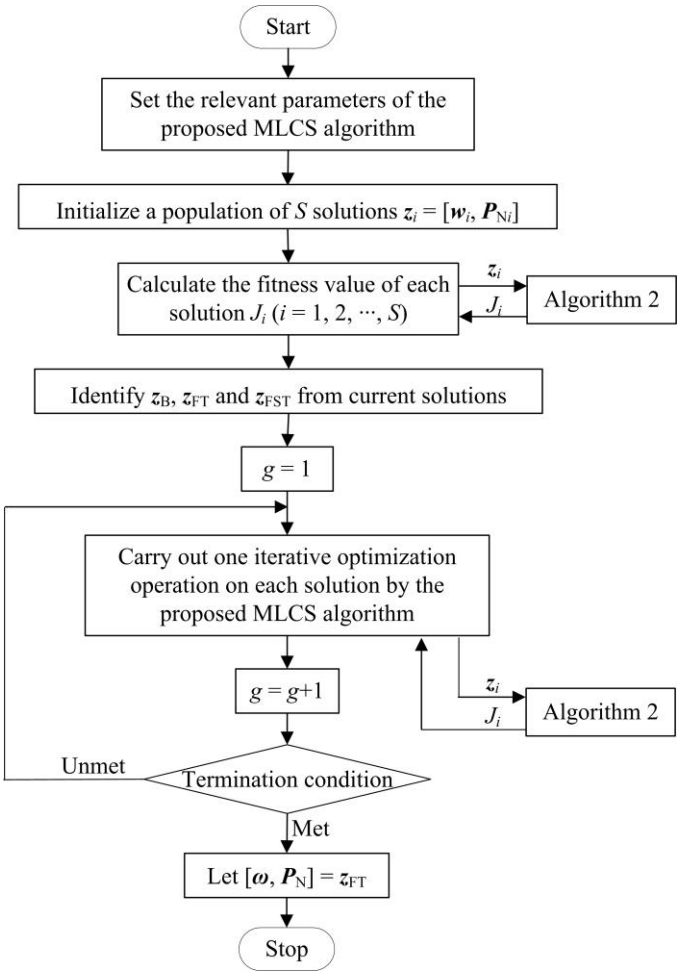


**Fig. 7.** Flowchart for solving the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_N$ using MLCS.

## 5. Experiments and analyses

In this section, 537 samples of production data from a 300 t LF within an iron & steel plant in China are employed to demonstrate the performance of the proposed FW-SLFN model, as well as the search capability of MLCS. Among these data, 437 samples are randomly chosen to develop the FW-SLFN model, and the rest are used to test the performance of this model.

The parameter settings of MLCS for the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_N$ in the proposed FW-SLFN model are the same as that used in Section 3.3.2, namely $\alpha = 0.01$, $p_a = 0.25$, $c_1 = 0.09$, $c_2 = 1.25$, $S = D$ (where $D = [8 + (8+1)\times N + (N+1)\times 1]$, and $N$ denotes the number of hidden neurons of the FW-SLFN model), and $g_{max} = 5000$.

For the design of SLFN-based models, determining the hidden neuron number ($N$) is a crucial step. One of the most frequently-used yet efficient ways is to choose $N$ by trial and error [51-53], which is also adopted by this paper. To be specific, firstly, according to our experience, the reasonable range of $N$ is assumed to be between six and 20. Then, trials are performed for each $N$ repeatedly 20 times on the 437 samples of modeling data, and the 6-fold (one subset has 72 samples of modeling data, and all the others have 73 samples of modeling data) cross-validation method is employed to seek out the optimal or near-optimal value of $N$, on the basis of the mean absolute error (MAE) criterion. Note that the activation function for the hidden layer of the FW-SLFN model is the sigmoid function and that for the output layer is the pure linear function. In addition, it should be noted that if not specified there are fifteen percent of modeling data employed for computing the early stopping condition during each modeling process involved in this study. Fig. 8 illustrates the trial results with regard to the MAE of 20 runs. It can be seen from Fig. 8 that the prediction performance of the FW-SLFN model is greatly influenced by the value of $N$, and when $N = 13$ the best result is achieved. Therefore, the topology of the FW-SLFN model is selected to be 8–13–1 in this study. After the topology is determined, all the 437 samples are utilized to determine $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ by the aforementioned method so as to obtain the proposed MSET prediction model (i.e. the FW-SLFN model), and 20 independent calculation runs are conducted to reduce random discrepancy. Thus, we can obtain 20 groups of predicted values of all samples – or 20 predicted values of each sample, and the mean predicted values of each sample are utilized for the following performance exhibition and comparison of the proposed FW-SLFN model unless special remark.

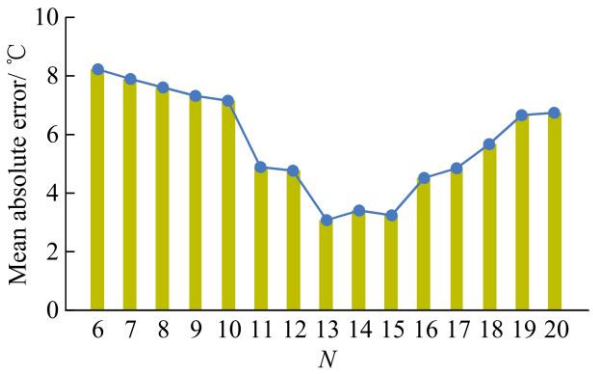


**Fig. 8.** Trial results with different *N*.

Fig. 9 (a) and Fig. 9 (b) respectively illustrate the results predicted with the developed FW-SLFN model on the 437 samples of modeling data and 100 samples of testing data. It can be seen that the model has high prediction accuracy and good generalization ability. Out of these results on the 100 samples of testing data, the absolute errors in 88% of the cases are lower than 5℃ (desirable value), in 94% of the cases they are lower than 7℃ (tolerable value), and only in 4% of the cases where the absolute errors are higher than 10℃. This demonstrates the effectiveness of the FW-SLFN model.

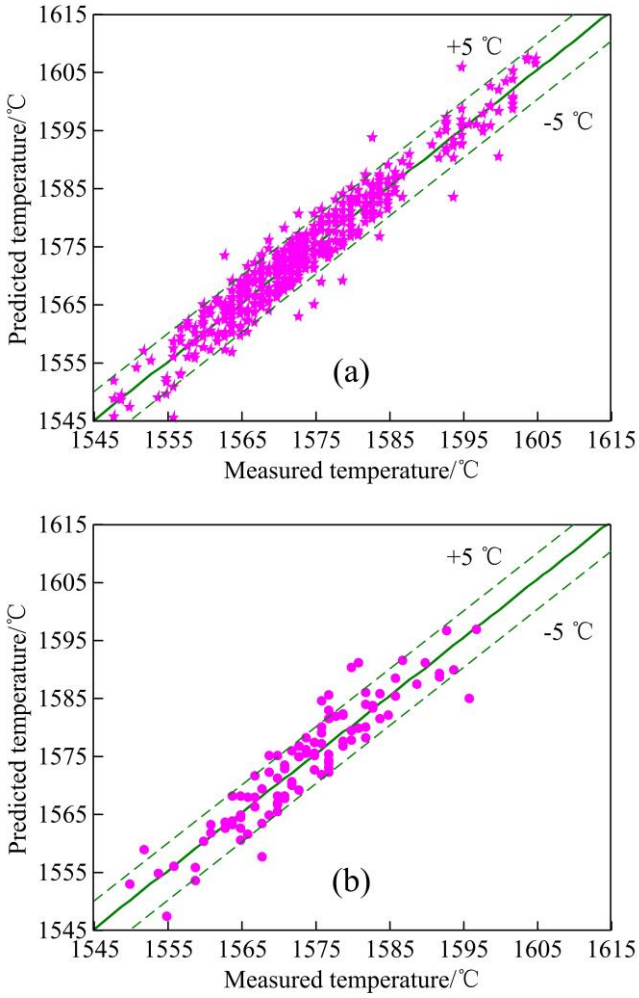


**Fig. 9.** Performance of the proposed FW-SLFN model on (a) the modeling data and (b) the testing data.

To demonstrate the superiority of the FW-SLFN model, this paper also develops another SLFN-based prediction model with the same features (i.e. inputs) and topology as that used in the FW-SLFN model but no features being weighted, hereafter referred to as the NFW-SLFN model, as the basis of comparison. To make the comparison fair, the parameters (i.e. the connection weights and thresholds) of the NFW-SLFN model is determined by MLCS with the same 437 samples. Besides, as with the proposed FW-SLFN model, 20 independent calculation runs are conducted and the mean predicted values are utilized for the performance exhibition and comparison of the NFW-SLFN model. Fig. 10 (a) and Fig. 10 (b) show its prediction results on the 437 samples of modeling data and 100 samples of testing data respectively. From Figs. 9 and

10, it can be found that both models could predict the MSET with good accuracy, while the prediction values given by the FW-SLFN model are much closer to the measured values than those given by the NFW-SLFN model. Also, it can be found that for both models the prediction accuracy on the testing data decreases when compared with that on the modeling data. The MAE value of the FW-SLFN model increases from 2.6883℃ on the modeling data to 3.0660℃ on the testing data, and that of the NFW-SLFN model increases from 3.2030℃ on the modeling data to 3.9530℃ on the testing data.

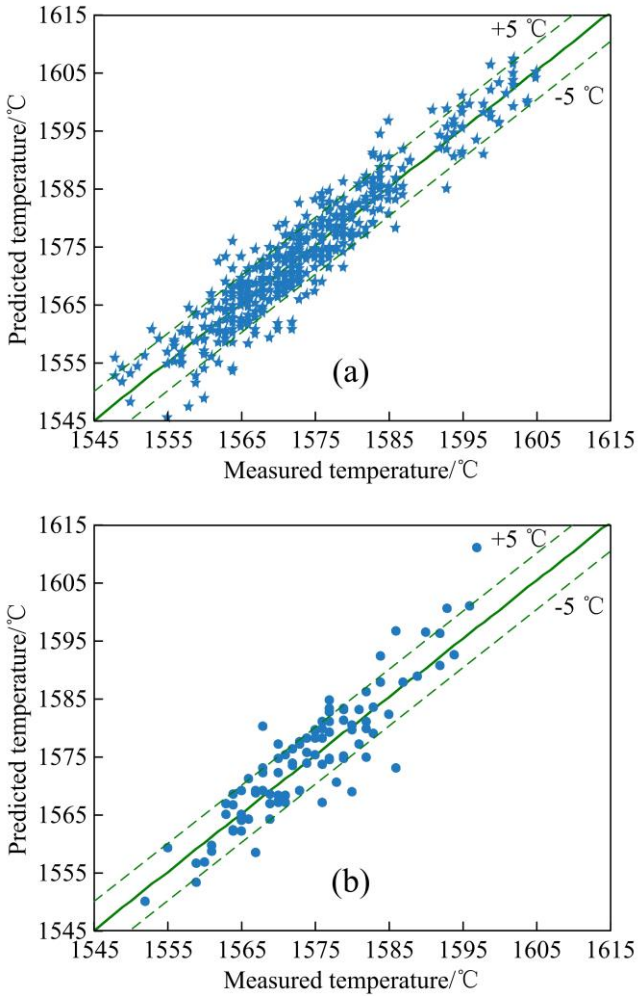


**Fig. 10.** Performance of the NFW-SLFN model on (a) the modeling data and (b) the testing data.

The prediction accuracy on the testing data is the important criterion for evaluating the performance of the MSET prediction model. For ease of comparison, the prediction errors (*PE*)

of the above two models on the 100 samples of testing data, as well as the differences between the prediction errors (*D_PE*) are respectively presented in Fig. 11(a) and Fig. 11(b). Here, the differences, *D_PE*, are the results obtained by subtracting the absolute prediction errors of the NFW-SLFN model from that of the FW-SLFN model. Thus, obviously the value of *D_PE* can be used to indicate whether the prediction performance of the FW-SLFN model is better than (i.e. $D\_PE < 0$), ties (i.e. $D\_PE = 0$), or worse than (i.e. $D\_PE > 0$) that of the NFW-SLFN model on the corresponding sample. Moreover, four indices are used for quantitative comparison of the performance of the two models on the testing data. They are the mean absolute error, root mean square error, mean relative error, and accuracy rate which is defined as

$$accuracy\ rate = \frac{N_a}{N_t} \cdot 100\% \tag{22}$$

where $N_a$ is the number of samples with absolute prediction error not higher than 5 °C, and $N_t$ is the number of total samples. The calculation results on the 100 samples of testing data in terms of these four indices for the two models are listed in Table 4. In Table 4, *MAE*, *RMSE*, *MRE*, and *AR* respectively denote the values of the mean absolute error, root mean square error, mean relative error, and accuracy rate, all of which are calculated on the basis of the mean predicted values; *MAE_B*, *RMSE_B*, *MRE_B*, and *AR_B* respectively denote the best (i.e. smallest) values of the mean absolute error, root mean square error and mean relative error, as well as the best (i.e. largest) value of the accuracy rate among the 20 runs; *MAE_W*, *RMSE_W*, *MRE_W*, and *AR_W* respectively denote the worst (i.e. largest) values of the mean absolute error, root mean square error and mean relative error, as well as the worst (i.e. smallest) value of the accuracy rate among the 20 runs. From Fig. 11 along with the data listed in Table 4, it is clear that the proposed FW-SLFN model performs better than the NFW-SLFN model. The *MAE*, *RMSE*, and *MRE* of the former are respectively decreased by 22.44%, 21.58%, and 22.42%, compared with those of the latter; whereas the *AR* of the former approaches 90%, an 11.39% improvement over the latter,

which demonstrates the excellent prediction performance of the FW-SLFN model from the practical point of view. In addition, it can be observed that the worst performance index values obtained from the 20 independent runs of the proposed FW-SLFN model are better than the best ones of the NFW-SLFN model. According to all these observations and comparisons above, it can be concluded that the feature-weighted strategy is necessary and effective for the prediction of the MSET in LF.

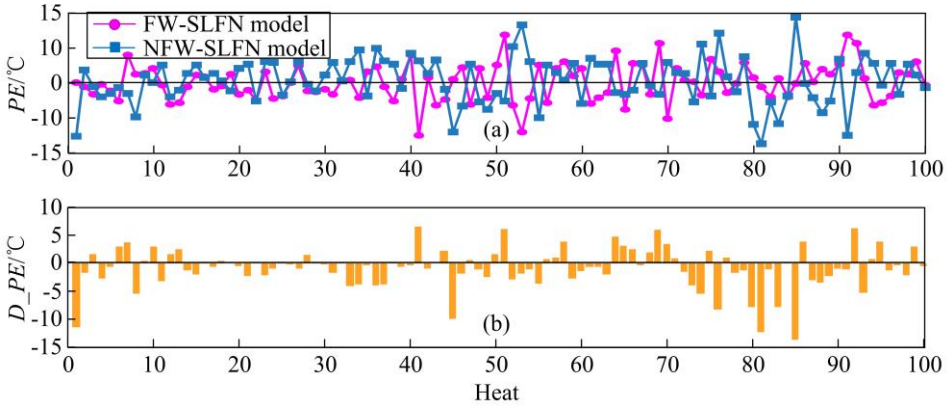


**Fig. 11.** Prediction error comparison on the testing data between the FW-SLFN and NFW-SLFN models.

**Table 4** Performance comparison on the testing data between the two prediction models.

| Model | Performance indices | | | | | |
|---|---|---|---|---|---|---|
| FW-SLFN model | *MAE* (℃) | 3.0660 | *MAE_B* (℃) | 2.9674 | *MAE_W* (℃) | 3.1982 |
| | *RMSE* (℃) | 3.8563 | *RMSE_B* (℃) | 3.6713 | *RMSE_W* (℃) | 4.1322 |
| | *MRE* (%) | 0.1948 | *MRE_B* (%) | 0.1886 | *MRE_W* (%) | 0.2033 |
| | *AR* (%) | 88 | *AR_B* (%) | 91 | *AR_W* (%) | 86 |
| NFW-SLFN model | *MAE* (℃) | 3.9530 | *MAE_B* (℃) | 3.5899 | *MAE_W* (℃) | 4.3129 |
| | *RMSE* (℃) | 4.9178 | *RMSE_B* (℃) | 4.6698 | *RMSE_W* (℃) | 5.4037 |
| | *MRE* (%) | 0.2511 | *MRE_B* (%) | 0.2281 | *MRE_W* (%) | 0.2741 |
| | *AR* (%) | 79 | *AR_B* (%) | 82 | *AR_W* (%) | 76 |

To further investigate the performance of MLCS, we compare it with ant colony optimization (ACO) [54], differential evolution (DE) [55], PSO [56], GA [57], and CS on the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_{\mathrm{N}}$ in the FW-SLFN model with the topology of 8–13–1 and the 437 samples of modeling data. For these algorithms, the parameter settings are obtained from their

original articles, except for $g_{max}$ and $S$. In this study, $g_{max}$ is set to 5000 for MLCS and CS, and 10000 for ACO, DE, PSO, and GA; and $S$ is set to 139 for the six algorithms. The reason that MLCS and CS both have half the $g_{max}$ value of ACO, DE, PSO or GA is that the number of fitness evaluation of MLCS or CS at each iteration is double that of ACO, DE, PSO or GA. The implementation of ACO, DE, PSO, GA or CS for the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_N$ is similar to that of MLCS, so we would not reiterate them here to save space. In addition, it should be pointed out that the aim here is comparing the search performance of the algorithms involved, so the termination condition is only $g > g_{max}$ and no samples need to be selected from the modeling data for computing the early stopping condition.

The results of the involved algorithms for the joint optimization of $\boldsymbol{w}$ and $\boldsymbol{P}_N$ within 20 repeated runs are presented in Table 5 (best results are shown in bold), where $F_{mean}$, $F_{min}$, $F_{max}$, and $SD$ respectively denote the mean, the minimum, the maximum, and the standard deviation of lowest fitness values achieved by an algorithm within 20 runs. We can find from Table 5 that CS and MLCS perform better than ACO, DE, PSO, and GA. It indicates that CS-based algorithms are relatively more suitable for optimizing $\boldsymbol{w}$ and $\boldsymbol{P}_N$ in the FW-SLFN model. Moreover, from the data in Table 5, we can also notice that MLCS has a better search accuracy than CS. Fig. 12 illustrates the convergence curves with respect to $F_{mean}$ of CS and MLCS on the joint optimization problem. It can be found that MLCS has higher search efficiency (namely convergence speed) and better search accuracy, which reveals again that MLCS has an evident improvement on CS.

**Table 5** Result comparison of MLCS, ACO, DE, PSO, GA, and CS on the joint optimization problem.

| Algorithm | Fitness value | | | |
|---|---|---|---|---|
| | $F_{mean}$ | $F_{min}$ | $F_{max}$ | *SD* |
| MLCS | **2.1897** | **1. 9717** | **3.1933** | **0.2987** |
| ACO | 6.9572 | 4.8544 | 9.8003 | 0.7194 |
| DE | 5.9575 | 3.9649 | 9.5716 | 1.3971 |
| PSO | 5.9632 | 3.9753 | 11.2785 | 2.0547 |
| GA | 6.3815 | 4.4058 | 11.8270 | 1.7913 |
| CS | 4.8957 | 3.5922 | 6. 6070 | 0.5612 |

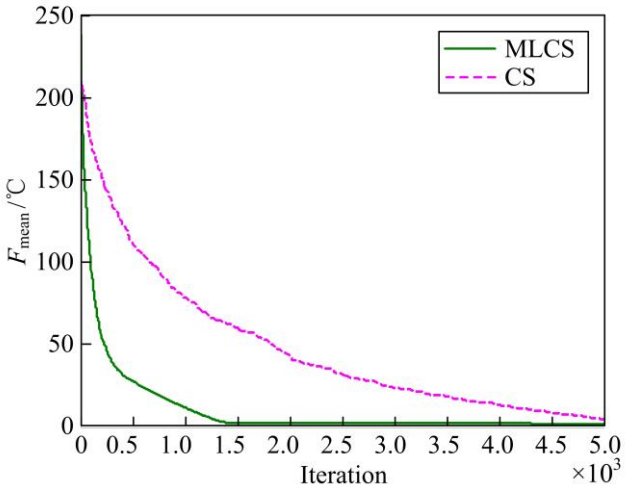


**Fig. 12.** Convergence curves of CS and MLCS on the joint optimization problem.

## 6. Conclusions

A novel feature-weighted SLFN-based model, optimized by a mutual learning CS (MLCS), is presented in this paper to predict the MSET in LF. Compared with existing models for predicting MSET in LF, the essential superiority of this model is its embodying the reality of LF refining processes where different features have different impacts on the MSET. So it is promising for improvement of prediction accuracy. Experimental studies are conducted based on actual production data from a 300 t LF in a Chinese iron & steel plant, and the results indicate the effectiveness of the presented prediction model and the necessity of the feature-weighted strategy.

Another main innovation of this paper is the construction of a new CS variant called mutual

learning CS (MLCS), which is employed to optimize the weight of each feature and the parameters of the SLFN so as to obtain the presented FW-SLFN model. One of the problems with CS as well as many of its variants is that the information (search experience) exchange among cuckoos is lacking during the search process, considerably impacting their search performance. Aiming at addressing this problem, a new mutual learning-based (ML-based) search strategy is proposed and employed in MLCS. Besides, a new bottom reinforcement learning-based (BRL-based) search strategy is also proposed and introduced in MLCS to further increase its capability. The search performance of MLCS has been tested on various benchmark functions as well as the above joint optimization problem. The results demonstrate that MLCS has superior search performance to its competitors on the considered problems in all aspects having been used for comparison, that is, the search accuracy, search efficiency (i.e. convergence speed), and search reliability. Despite its promising search performance, our MLCS still has limitations. First of all, compared with CS, two more parameters (i.e. $c_1$ and $c_2$) are used by the algorithm to perform the two new search strategies. Consequently, the parameter tuning process used to achieve a reasonably good performance of MLCS becomes time consuming. As for the two common parameters of MLCS and CS (i.e. $\alpha$ and $p_a$), our current study sets them directly according to the recommendation of Yang and Deb [18, 42]. There may be better value combinations of the four parameters. But their tuning process will no doubt become much more time consuming, and it might also require retuning when the algorithm is applied to solve different optimization problems.

Based on the current study, several future work directions can be pursued. Firstly, a parameter self-learning mechanism could be constructed according to some real-time running indices of the algorithm (e.g. the number or the degree of the improved solutions) so as to adaptively tune the involved four parameters. Secondly, there is still room for improving the exemplar generation mechanism. The generation mechanism used in this paper is a kind of blindness; therefore a

more effective mechanism is worthy to research. Thirdly, with the development of artificial intelligence, various nature-inspired optimization algorithms are constantly emerging. By combining the feature-weighted modeling idea with some other excellent nature-inspired optimization algorithms (e.g. firefly algorithm [58]) would be helpful for the performance enhancement of the developed prediction model. Finally, the proposed feature-weighted modeling method could be also applied to other LF refining processes or other similar complex industrial processes.

## Acknowledgement

This work was supported by the Fundamental Research Funds for the Central Universities [grant number N162504013], and the National Key Research and Development Program of China [grant number 2017YFB0304203].

Declarations of interest: none.

## References

[1] Ü. Çamdali, M. Tunç, Steady state heat transfer of ladle furnace during steel production process, J. Iron Steel Res. Int., 13 (2006) 18-20,25.
[2] O. Volkova, D. Janke, Modelling of temperature distribution in refractory ladle lining for steelmaking, ISIJ Int., 43 (2003) 1185-1190.
[3] K. Feng, D.F. He, A.J. Xu, H.B. Wang, End temperature prediction of molten steel in LF based on CBR-BBN, Steel Res. Int., 87 (2016) 79-86.
[4] W. Lv, Z.Z. Mao, P. Yuan, M.X. Jia, Multi-kernel learnt partial linear regularization network and its application to predict the liquid steel temperature in ladle furnace, Knowl-Based Syst., 36 (2012) 280-287.
[5] Y.J. Wu, Z.H. Jiang, M.F. Jiang, W. Gong, D.P. Zhan, Temperature prediction model of molten steel in LF, J. Iron Steel Res., 14 (2002) 9-12 (in Chinese).
[6] N.K. Nath, K. Mandal, A.K. Singh, B. Basu, C. Bhanu, S. Kumar, A. Ghosh, Ladle furnace on-line reckoner for prediction and control of steel temperature and composition, Ironmak. Steelmak., 33 (2006) 140-150.
[7] W. Lü, Z.Z. Mao, P. Yuan, Ladle furnace liquid steel temperature prediction model based on optimally pruned bagging, J. Iron Steel Res. Int., 19 (2012) 21-28.
[8] G.M. Mendez, A. Cavazos, R. Soto, L.A. Leduc Lezama, Entry temperature prediction of a hot strip mill by a hybrid learning type-2 FLS, Journal of Intelligent & Fuzzy Systems, 17 (2006) 583-596.
[9] Y.G. Sun, D.X. Wang, B.S. Tao, T. Yan, Y. Shi, S.B. Fang, Y.H. Wang, An intelligent ladle furnace control system, in: The 3rd World Congress on Intelligent Control and Automation, IEEE, Hefei, China, 2000, pp. 330-334.
[10] H.X. Tian, Z.Z. Mao, A.N. Wang, A new incremental learning modeling method based on multiple models for temperature prediction of molten steel in LF, ISIJ Int., 49 (2009) 58-63.
[11] H.X. Tian, Z.Z. Mao, An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace, IEEE Trans. Autom. Sci. Eng., 7 (2010) 73-80.
[12] X.J. Wang, M.S. You, Z. Mao, P. Yuan, Tree-Structure Ensemble General Regression Neural Networks applied to predict the molten steel temperature in Ladle Furnace, Adv. Eng. Inform., 30 (2016) 368-375.

[13] F. He, A.J. Xu, H.B. Wang, D.F. He, N.Y. Tian, End temperature prediction of molten steel in LF based on CBR, Steel Res. Int., 83 (2012) 1079-1086.
[14] A. Saha, S. Das, Categorical fuzzy k-modes clustering with automated feature weight learning, Neurocomputing, 166 (2015) 422-435.
[15] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks, 2 (1989) 359-366.
[16] C. Anitescu, E. Atroshchenko, N. Alajlan, T. Rabczuk, Artificial neural network methods for the solution of second order boundary value problems, Computers, Materials & Continua, 59 (2019) 345-359.
[17] H.X. Tian, Z.Z. Mao, Y. Wang, Hybrid modeling of molten steel temperature prediction in LF, ISIJ Int., 48 (2008) 58-62.
[18] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE, Coimbatore, India, 2009, pp. 210-214.
[19] M. Shehab, A.T. Khader, M.A. Al-Betar, A survey on applications and variants of the cuckoo search algorithm, Appl. Soft Comput., 61 (2017) 1041-1059.
[20] X. Zhu, N. Wang, Splicing process inspired cuckoo search algorithm based ENNs for modeling FCCU reactor-regenerator system, Chemical Engineering Journal, 354 (2018) 1018-1031.
[21] P. Ong, Z. Zainuddin, Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction, Appl. Soft Comput., 80 (2019) 374-386.
[22] X. Du, J. Wang, V. Jegatheesan, G. Shi, Parameter estimation of activated sludge process based on an improved cuckoo search algorithm, Bioresour. Technol., 249 (2018) 447-456.
[23] W. Sun, J. Sun, Daily PM2.5 concentration prediction based on principal component analysis and LSSVM optimized by cuckoo search algorithm, Journal of environmental management, 188 (2017) 144-152.
[24] M.A. Sanchez, O. Castillo, J.R. Castro, Information granule formation via the concept of uncertainty-based information with Interval Type-2 Fuzzy Sets representation and Takagi–Sugeno–Kang consequents optimized with Cuckoo search, Appl. Soft Comput., 27 (2015) 602-609.
[25] J. Zhao, P.K. Wong, Z. Xie, X. Ma, X. Hua, Design and control of an automotive variable hydraulic damper using cuckoo search optimized pid method, International Journal of Automotive Technology, 20 (2019) 51-63.
[26] A. Sikander, P. Thakur, R.C. Bansal, S. Rajasekar, A novel technique to design cuckoo search based FOPID controller for AVR in power systems, Computers and Electrical Engineering, 70 (2018) 261-274.
[27] B. Yang, J. Miao, Z. Fan, J. Long, X. Liu, Modified cuckoo search algorithm for the optimal placement of actuators problem, Appl. Soft Comput., 67 (2018) 48-60.
[28] S. Afanasyeva, J. Saari, O. Pyrhönen, J. Partanen, Cuckoo search for wind farm optimization with auxiliary infrastructure, Wind Energy, 21 (2018) 855-875.
[29] M. Mareli, B. Twala, An adaptive Cuckoo search algorithm for optimisation, Applied Computing and Informatics, 14 (2018) 107-115.
[30] G.B. Murali, B. Deepak, B. Biswal, G.B. Mohanta, A. Rout, Robotic optimal assembly sequence using improved cuckoo search algorithm, Procedia Computer Science, 133 (2018) 323-330.
[31] G.D. Goez-Sanchez, J.A. Jaramillo-Garzon, R.A. Velasquez, Performance comparison of particle swarm optimization and Cuckoo search for online route planning, IEEE Aerospace and Electronic Systems Magazine, 33 (2018) 40-50.
[32] T.T. Nguyen, D.N. Vo, B.H. Dinh, An effectively adaptive selective cuckoo search algorithm for solving three complicated short-term hydrothermal scheduling problems, Energy, 155 (2018) 930-956.
[33] M.A. Mohamed, A.M. Eltamaly, A.I. Alolah, A.Y. Hatata, A novel framework-based cuckoo search algorithm for sizing and optimization of grid-independent hybrid renewable energy systems, International Journal of Green Energy, 16 (2019) 86-100.
[34] X. Meng, J. Chang, X. Wang, Y. Wang, Multi-objective hydropower station operation using an improved cuckoo search algorithm, Energy, 168 (2019) 425-439.
[35] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artif. Intell. Rev., 39 (2013) 315-346.
[36] A.K. Bhateja, A. Bhateja, S. Chaudhury, P.K. Saxena, Cryptanalysis of vigenere cipher using cuckoo search, Appl. Soft Comput., 26 (2015) 315-324.
[37] L. Huang, S. Ding, S. Yu, J. Wang, K. Lu, Chaos-enhanced cuckoo search optimization algorithms for global optimization, Appl. Math. Model., 40 (2016) 3860-3875.
[38] M.K. Naik, R. Panda, A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition, Appl. Soft Comput., 38 (2016) 661-675.
[39] E. Valian, S. Tavakoli, S. Mohanna, A. Haghi, Improved cuckoo search for reliability optimization problems, Comput. Ind. Eng., 64 (2013) 459-468.
[40] L.J. Wang, Y.W. Zhong, Y.L. Yin, Nearest neighbour cuckoo search algorithm with probabilistic mutation, Appl. Soft Comput., 49 (2016) 498-509.
[41] N.J. Cheung, X.M. Ding, H.B. Shen, A nonhomogeneous cuckoo search algorithm based on quantum

mechanism for real parameter optimization, IEEE transactions on cybernetics, 47 (2017) 391-402.
[42] X.S. Yang, Nature-Inspired Optimization Algorithms, 1st ed., Elsevier, Oxford, 2014.
[43] P. Sekhar, S. Mohanty, An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement, Int. J. Elec. Power, 75 (2016) 303-310.
[44] R.N. Mantegna, Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes, Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics, 49 (1994) 4677-4683.
[45] X. Yao, Y. Liu, G.M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput., 3 (1999) 82-102.
[46] W.H. Lim, N.A. Mat Isa, Adaptive division of labor particle swarm optimization, Expert Syst. Appl., 42 (2015) 5887-5903.
[47] W.H. Lim, N.A. Mat Isa, Two-layer particle swarm optimization with intelligent division of labor, Eng. Appl. Artif. Intel., 26 (2013) 2327-2348.
[48] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 2nd ed. ed., Chapman & Hall/CRC, Boca Raton, 2000.
[49] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics, 15 (2009) 617-644.
[50] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput., 12 (2008) 107-125.
[51] A. Saptoro, H.M. Yao, M.O. Tadé, H.B. Vuthaluru, Prediction of coal hydrogen content for combustion control in power utility using neural network approach, Chemom. Intell. Lab. Syst., 94 (2008) 149-159.
[52] A. Barati-Harooni, A. Najafi-Marghmaleki, A.H. Mohammadi, A reliable radial basis function neural network model (RBF-NN) for the prediction of densities of ionic liquids, J. Mol. Liq., 231 (2017) 462-473.
[53] H. Guo, X. Zhuang, T. Rabczuk, A deep collocation method for the bending analysis of Kirchhoff plate, Computers, Materials & Continua, 59 (2019) 433-456.
[54] S. Bououden, M. Chadli, H.R. Karimi, An ant colony optimization-based fuzzy predictive control approach for nonlinear processes, Inform. Sciences, 299 (2015) 143-158.
[55] G. Quaranta, G.C. Marano, R. Greco, G. Monti, Parametric identification of seismic isolators using differential evolution and particle swarm optimization, Appl. Soft Comput., 22 (2014) 458-464.
[56] M.A. Rahman, S. Anwar, A. Izadian, Electrochemical model parameter identification of a lithium-ion battery using particle swarm optimization method, J. Power Sources, 307 (2016) 86-97.
[57] C. Chisari, C. Bedon, C. Amadio, Dynamic and static identification of base-isolated bridges using genetic algorithms, Eng. Struct., 102 (2015) 80-92.
[58] S. Chakraborty, N. Dey, S. Samanta, A.S. Ashour, V.E. Balas, Firefly algorithm for optimized nonrigid demons registration, in: X.S. Yang, J.P. Papa (Eds.) Bio-Inspired Computation and Applications in Image Processing, Academic Press, 2016, pp. 221-237.